KETTERÄT KEHITYSMENETELMÄT SÄHKÖISEN ASIAKIRJAJÄRJESTELMÄN SUUNNITTELUSSA

Jarmo Tikka

Tampereen yliopisto
Informaatiotieteiden yksikkö
Informaatiotutkimus ja interaktiivinen media
Pro gradu -tutkielma
Tammikuu 2013

TAMPEREEN YLIOPISTO, Informaatiotieteiden yksikkö

Informaatiotutkimuksen ja interaktiivisen median maisteriohjelma

TIKKA, JARMO: Ketterät kehitysmenetelmät sähköisen asiakirjajärjestelmän suunnittelussa

Pro gradu -tutkielma, 99 s.

Tammikuu 2013

Tutkimuksessa käsiteltiin ketterien kehitysmenetelmien soveltamista sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun. Ketterät kehitysmenetelmät ovat erityisesti tuotesuunnittelussa ja tietojenkäsittelyssä käytettyjä suunnittelumenetelmiä, jotka lisäävät järjestelmän, tuotteen tai palvelun kehittämisen nopeutta ja joustavuutta. Tässä työssä menetelmistä käsiteltiin osittamista, pilottiprojekteja, jatkuvaa testaamista, prototyypittämistä ja iterointia. Työssä tutkittiin, miten sähköisten asiakirjajärjestelmien tietosisällön suunnittelua voidaan nopeuttaa ja tehdä joustavammaksi ketterien kehitysmenetelmien avulla. Joustavuudeksi laskettiin paitsi mahdollisuudet palata suunnittelussa aiempiin suunnitteluvaiheisiin, myös suunnittelun käyttäjälähtöisyys. Lisäksi pohdittiin, kuinka tietosisällön ketterämpi suunnittelu vaikuttaa kokemattomien suunnittelijoiden kykyyn hallita suunnittelua.

Ketterien kehitysmenetelmien soveltamista tutkittiin luomalla olemassa olevien asiakirjajärjestelmien suunnittelumallien pohjalta uusi sähköisen asiakirjajärjestelmän tietosisällön suunnittelumalli, joka hyödyntää ketteriä kehitysmenetelmiä. Malli kehitettiin etsimällä eri tieteenalojen kirjallisuudesta esimerkkejä ketterien kehitysmenetelmien soveltamisesta järjestelmien suunnittelussa ja soveltamalla näitä sovellustapoja malliin, jos ne lisäsivät suunnitteluun nopeutta ja joustavuutta. Mallin luomisen jälkeen pohdittiin kirjallisuuden pohjalta, kuinka malli vaikuttaa kokemattomien suunnittelijoiden mahdollisuuksiin hallita suunnittelua. Kirjallisuuden lisäksi työssä hyödynnettiin kirjoittajan omia kokemuksia asiakirjajärjestelmien suunnittelusta. Mallia ei testattu käytännössä.

Mallin rakentaminen osoitti, että ketterän kehityksen menetelmiä on mahdollista mielekkäästi soveltaa sähköisen asiakirjajärjestelmän tietosisällön suunnittelussa. Tarkastelun pohjalta on perusteltua olettaa, että ketterillä kehitysmenetelmillä voidaan lisätä sähköisen asiakirjajärjestelmän suunnittelun nopeutta ja joustavuutta. Ketterän kehityksen menetelmät ovat erityisen lupaavia suunnittelun käyttäjälähtöisyyden lisäämisessä, koska niiden avulla käyttäjät voidaan tuoda aktiivisemmin mukaan suunnittelutyöhön.

Suunnittelun hallittavuutta kokemattomien suunnittelijoiden kannalta lisäävät mallissa erityisesti lisääntynyt yhteys käyttäjiin, mahdollisuus suunnittelun peilaamiseen palautteen avulla ja iteratiivisuuden mahdollistama vaiheittainen oppiminen. Myös projektin jakautuminen pienempiin kokonaisuuksiin ja asteittainen suunnittelun kehittäminen edistävät hallittavuutta. Hallittavuutta voi kuitenkin vähentää se, että lisääntynyt yhteys käyttäjiin edellyttää suunnittelijoilta parempia kommunikaatio- ja vuorovaikutustaitoja. Lisäksi käyttäjälähtöisyys pakottaa suunnittelijat jakamaan valtaa käyttäjien kanssa suunnittelupäätösten tekemisessä. Ketterän kehityksen menetelmät lisäävät myös suunnittelun monimutkaisuutta pilkkomalla suunnittelua pienempiin kokonaisuuksiin.

Avainsanat: asiakirjahallinta, informaatiojärjestelmien suunnittelu, sähköiset asiakirjajärjestelmät, ketterä kehitys, kokemattomuus suunnittelussa

Sisällysluettelo

1 JOHDANTO

Sähköisten asiakirjojen yleistyttyä niiden hallinnassa on alettu yleisesti käyttää sähköisiä asiakirjajärjestelmiä. Sähköisellä asiakirjajärjestelmällä tarkoitan työssä ohjelmistoa, jolla hallitaan organisaation sähköisiä asiakirjoja ja johon organisaation sähköiset asiakirjat siirretään muista käytetyistä ohjelmistoista. Ajatuksena sähköisten asiakirjajärjestelmien käytössä on ollut, että organisaatiot keskittävät asiakirjojen hallinnan yhteen tietojärjestelmään, jossa asiakirjoja voidaan hallita luotettavammin. Sähköisten asiakirjajärjestelmien käyttöön perustuva malli on kuitenkin ollut monelta osin pettymys. Asiakirjajärjestelmäprojektien toteuttamisessa on ollut paljon ongelmia ja ne ovat epäonnistuneet usein (ks. esim. Gunnlaugsdottir 2008). Epäonnistuminen järjestelmien suunnittelussa on ollut seurausta paitsi sähköisten asiakirjajärjestelmien suunnittelun vaikeudesta myös siitä, että yleensäkin informaatiojärjestelmien toteutusprojektit epäonnistuvat todella usein, eivätkä saavuta asetettuja tavoitteita (Remenyi & Sherwood-Smith 1999, 16).

Asiakirjajärjestelmien suunnittelun ja käyttöönoton (implementation) ongelmista huolimatta asiakirjahallinnan tutkimuksessa on käsitelty vähän ongelmien syitä. Harvoissa asiaa tarkastelleissa tutkimuksissa, kuten Johanna Gunnlaugsdottirin (2006, 2008) kirjoituksissa, on etsitty laajempia selittäviä tekijöitä epäonnistumisille. Nämä tutkimukset ovat ansiokkaasti eritelleet ongelmia, jotka selittävät projektien epäonnistumisia. Tutkimuksissa on kuitenkin pääosin sivuutettu suunnittelijoiden näkökulma sähköisten asiakirjajärjestelmien kehittämisessä, eikä ole pohdittu, miten sähköisten asiakirjajärjestelmien suunnitteluprosessia voitaisiin tehostaa ja tehdä hallittavammaksi.

Asiakirjahallinnan tutkimuksissa ja ohjeistuksissa ei myöskään ole käsitelty sitä, miten suunnittelijoiden oppimista projektin toteuttamisesta voitaisiin tukea. Asiakirjajärjestelmien suunnitteluun liittyvät ohjeistukset, kuten arkistolaitoksen AMS-opas (Kansallisarkisto 2012) ja eAMS-käyttöönottosuunnitelma (Arkistolaitos 2010), lähtevät oletuksesta, että projektiryhmällä on jo projektin alussa kaikki suunnittelussa tarvittava osaaminen tai heitä koulutetaan ennen projektia tarvittavan osaamisen kehittämiseksi. Ajatuksena on, että suunnitteluryhmään hankitaan valmiiksi tarvittava osaaminen ennen projektin alkua.

Todellisuudessa suunnittelijat eivät kuitenkaan ole niin kokeneita ja osaavia kuin asiakirjahallinnan ohjeistukset olettavat. Esimerkiksi informaatiojärjestelmien suunnittelua käsittelevissä tutkimuksissa suunnittelijoiden suunnittelun aikana tapahtuvan oppimisen keskeisyys tunnistetaan jo paremmin (ks. esim. Evans & Rouche 2004; Bansler & Havn 2010). Asiakirjahallinnan tutkimuksessa ja ohjeistuksissa tätä näkökulmaa ei kuitenkaan ole yleensä huomioitu. Suunnittelijoiden kehittymisen huomioimattomuus on erityisen ongelmallista yhdistettynä siihen, että sähköisten asiakirjajärjestelmien suunnittelun ohjeistukset perustuvat edelleen pääosin lineaarisiin suunnittelumalleihin, joissa järjestelmän suunnittelu toteutetaan yhtenä laajana suoraviivaisena vaiheesta vaiheeseen etenevänä projektina.

Ajatus kaiken tarvittavan osaamisen hankkimisesta ennen suunnittelun aloittamista ja järjestelmän suunnittelu laajana lineaarisena projektina vaikuttaa selkeältä ja tehokkaalta lähestymistavalta, mutta ei välttämättä ole sitä. Olen itse ollut suunnittelemassa sähköisen asiakirjajärjestelmän tietosisältöä ja arkistonmuodostussuunnitelmaa Itä-Suomen yliopistolle ja kirjoittanut kokemusteni pohjalta tutkielman (Tikka 2010), jossa käsittelen laajaan asiakirjajärjestelmäprojektiin liittyviä haasteita. Projektissa suunnittelu toteutettiin lineaarisesti ja tällä tavalla toteutettuna suunnittelun hallinta oli kokemattomalle projektiryhmälle erittäin haastavaa.

Ensinnäkin lineaarisilla menetelmillä toteutettu suunnittelutyö jakaantuu niin suuriin kokonaisuuksiin, että niiden hallitseminen on vaikeaa. Toiseksi lineaarisuus aiheuttaa sen, että suunnittelu pitäisi pystyä toteuttamaan kerralla oikein. Lineaarisissa malleissa perusajatus on, että edellisiä vaiheita ei toteutettaisi enää uudestaan, kun ne on kerran tehty, vaan jokainen vaihe suoritetaan kerran ja siirrytään sen jälkeen eteenpäin. Tästä ohjeistusten esittämästä suoraviivaisuudesta tehdään toki käytännön suunnittelussa poikkeuksia, mutta varsinaisiin malleihin ei kuitenkaan yleensä sisälly järjestelmällisiä paluupolkuja aiempiin suunnittelun vaiheisiin. Pyrkimys suunnittelun toteuttamiseen kerralla oikein aiheuttaa sen, että tehdyt virheet kertautuvat helposti loppua kohti ja niitä ei pystytä korjaamaan tehokkaasti.

Kolmas keskeinen ongelma lineaarisessa suunnittelussa liittyy kokemusteni pohjalta siihen, että palautteen kerääminen käyttäjiltä ja heidän tuomisensa mukaan suunnitteluun on vaikeaa, koska suunnittelun tuloksena syntyy toimivia testijärjestelmiä vasta aivan projektin loppuvaiheessa. Käyttäjät joutuvat lineaarisessa kehittämisessä hahmottamaan järjestelmien toimintaa abstraktien järjestelmäkuvausten ja irrallisten järjestelmän osien pohjalta. Tämän takia he eivät pysty hahmottamaan järjestelmän todellista toimintaa ja vaikutusta työskentelyynsä ja

antamaan tehokkaasti palautetta. (Alavi 1984, 557) Lineaarisuuden takia suunnittelijan kyky oppia, saada palautetta ja muokata järjestelmää vaiheittain paremmaksi ovat rajalliset ja virheiden korjaaminen jälkikäteen vaikeaa ja kallista.

Suunnittelijoiden näkökulman vähäisyyden lisäksi asiakirjahallinnan tutkimuksessa ei ole juuri käsitelty muilla tieteenaloilla esitettyjä keinoja tehostaa suunnittelutyötä. Tässä työssä etsin inspiraatiota sähköisen asiakirjajärjestelmän suunnitteluun muilta tieteenaloilta. Luon sähköisen asiakirjajärjestelmän suunnitteluun vaihtoehtoisen mallin, jossa hyödynnetään muiden tieteenalojen keskustelua projektien toteuttamisesta ja informaatiojärjestelmien suunnittelusta.

Käytän suunnittelumallin kehittelyssä niin sanottuja ketteriä kehitysmenetelmiä. Ketterät kehitysmenetelmät ovat alun perin tuotesuunnittelun ja tietojenkäsittelyn tutkimuksessa kehitettyjä menetelmiä, joilla pyritään lisäämään tuotteen tai järjestelmän suunnitteluun nopeutta ja joustavuutta. Ketterien kehitysmenetelmien tutkiminen ja soveltaminen on laajentunut tuotesuunnittelusta ja ohjelmistosuunnittelusta muun muassa informaatiojärjestelmien suunnitteluun ja hallintotieteisiin (ks. esim. Bansler & Havn 2010; Leybourne 2009).

Ketterien kehitysmenetelmien avulla pyrin tekemään suunnittelumallista nopeamman ja joustavamman verrattuna perinteisiin sähköisten asiakirjajärjestelmien suunnittelumalleihin. Joustavuuden kriteerinä korostan erityisesti suunnittelun käyttäjälähtöisyyden lisäämistä. Lisäksi pohdin työssä luomani suunnittelumallin pohjalta ketterien kehitysmenetelmien hyödyntämisen vaikutusta suunnittelun hallittavuuteen. Hallittavuuden tarkastelussa näkökulmana on kokemattoman suunnittelijaryhmän kyky hallita suunnittelua. Suunnittelijoiden kokemattomuuden valinta näkökulmaksi perustuu omiin kokemuksiini asiakirjajärjestelmän suunnittelusta ja havaintoihini siitä, että asiaa on käsitelty asiakirjahallinnan kirjallisuudessa vähän.

2 AIEMPI TUTKIMUS

2.1 Ketterän kehityksen perusperiaatteet

Kuten ketterän kehittämisen useimmin käytetty englanninkielinen nimi rapid development kertoo, ketterässä kehittämisessä nopeus on keskeistä. Ketterän kehityksen nopeudella tarkoitetaan yleensä nopeutta, jolla suunnittelussa saavutetaan tuloksia. On kuitenkin tärkeää huomata, että tämä tulosten saavuttamisen nopeus tarkoittaa paitsi lopullisen järjestelmän valmiiksi saamista, myös erilaisten välitavoitteiden saavuttamista, joiden perusteella voidaan arvioida suunnittelun etenemistä. Suunnittelun nopeuttamisen lisäksi ketterässä kehittämisessä pyritään myös lisäämään suunnittelun joustavuutta. Joskus englanninkielisessä keskustelussa erotetaan joustavuutta korostava kehittäminen nopeutta korostavasta käyttämällä termiä agile development, mutta useimmin puhutaan vain rapid developmentista, joka sisältää myös pyrkimyksen suunnittelun joustavuuteen. Kun välitavoitteita saavutetaan usein ja kehittämissyklit ovat lyhyitä, voidaan helpommin muuttaa ja suunnata suunnittelua niin, että muuttuvat tarpeet ja havaitut ongelmat voidaan huomioida. Tähän nopeaan tavoitteiden saavuttamiseen ja joustavuuteen voidaan päästä soveltamalla monia erilaisia menetelmiä. (Jain & Chandrasekaran 2009, 30–32)

Ketterät kehitysmenetelmät voidaan määritellä lukuisin eri tavoin, koska ne ovat vain laaja joukko erilaisia menetelmiä, joilla tuotteen tai järjestelmän suunnittelusta pyritään tekemään ”ketterämpää”. Ketteryyden käsite tarkoittaa eri asioita riippuen määrittelijästä ja tieteenalasta. Goldmanin et al. (1995) mukaan ketteryys tarkoittaa suunnittelussa muun muassa dynaamisuutta, kontekstin huomioimista ja muutoksiin varautumista. Jainin ja Chandrasekaran (2009, 30) mukaan ketterä järjestelmäkehitys tarkoittaa informaatiojärjestelmien suunnittelun yhteydessä minkä tahansa menetelmän, työkalun tai tekniikan hyödyntämistä, joka tuo suunnitteluprosesseihin nopeutta ja parantaa onnistumisen mahdollisuuksia. Leybornen (2009, 523) mielestä ketterien menetelmien tavoitteena on luoda mukautuvia tuotteita, joita on helppoa ja halpaa muokata muuttuvien tarpeiden mukaan. Tavoitteena on vähentää lineaariseen malliin pohjautuvan projektinhallinnan ongelmia, kuten rutiiniluonteista toteutustapojen soveltamista ja tiukkaa suunnitelmallisuutta ja korostaa sen sijaan erilaisten vaihtoehtojen rohkeaa kokeilemista, mikä edistää ihmisten oppimista ja innovoimista (Leyborne 2009, 523–524).

Asiakirjahallinnan tutkimuksessa ketterien kehitysmenetelmien soveltamista on käsitelty vain satunnaisesti. Asiakirjahallinnasta käydyssä keskustelussa paljon huomiota saaneen kontinuumimallin kehittäjä Frank Upward (2000) on tosin jo vuosituhannen vaihteessa kritisoinut asiakirjajärjestelmien suunnittelussa käytettyjä lineaarisia, vaiheittain eteneviä malleja, jotka pohjautuvat vanhentuneeseen paradigmaan informaatiojärjestelmien analysoinnista. Upwardin mukaan järjestelmien suunnittelu tällaisen niin sanotun järjestelmien kehittämisen elinkaarimallin pohjalta on epärealistinen. Hänen mukaansa järjestelmiä pitäisi kehittää asiakirjahallinnan kontinuumimalliin sidoksissa olevien lähestymistapojen, kuten ketterien kehitysmenetelmien, olioperustaisten ympäristöjen, jatkuvan kehittämisen ja prototyypittämisen avulla. (Upward 2000, 126–127)

Vaikka asiakirjahallinnassa ketteriä kehitysmenetelmiä ei ole juuri huomioitu, monen muun alan kirjallisuudessa ne ovat keskeisiä. Erityisesti tuotesuunnittelussa, ohjelmistojen kehittämisessä ja informaatiojärjestelmien suunnittelussa ketterät kehitysmenetelmät ovat saaneet paljon huomiota. Tuotesuunnittelussa ketteriä kehitysmenetelmiä on sovellettu informaatiojärjestelmiä kauemmin ja ketterien kehitysmenetelmien käytöstä on julkaistu alalla paljon kirjallisuutta (ks. esim. Liou 2008; Xie & Tu 2011). Tuotesuunnittelun tutkimukset ovat kuitenkin lähestymistavoiltaan kaukana omasta rajauksestani, eikä niiden soveltaminen ole mielekästä. Myös ketteriä kehitysmenetelmiä käytännönläheisesti ohjelmistojen suunnitteluun ja toteuttamiseen soveltavat tutkimukset ovat omalle ongelmanasettelulleni useimmiten etäisiä (ks. esim. McConnell 1996; Malcolm 2001). Vaikka näissä tutkimuksissa on käsitelty ketterän kehityksen periaatteita ja yleisiä ajatusmalleja niiden taustalla, ne ovat silti lähestymistavaltaan kaukana sähköisen asiakirjajärjestelmän kaltaisten informaatiojärjestelmien suunnittelusta.

Tietojenkäsittelytieteissä on kuitenkin myös toinen tutkimussuuntaus, jota kutsutaan rapid system developmentiksi (RSD). RSD:ssä tutkitaan, miten ketteriä kehitysmenetelmiä voidaan soveltaa erilaisten informaatiojärjestelmien suunnittelussa. (Jain & Chandrasekaran 2009, 30) Informaatiojärjestelmien suunnittelua ja käyttöönottoa käsittelevät tutkimukset liikkuvat yleensä tietojenkäsittelytieteiden ja hallintotieteiden välimaastossa. Toiset tutkimukset keskittyvät enemmän informaatiojärjestelmien suunnittelun hallintaan ja toiset painottavat enemmän ohjelmistokehityksessä käytettyjen ketterien kehitysmenetelmien soveltamista informaa-

tiojärjestelmien suunnittelussa. Molemmat näkökulmat ovat hyödyllisiä sähköisen asiakirjajärjestelmän tietosisällön suunnittelumallin luomisessa.

Vaikka tuotesuunnittelu- ja tietojärjestelmätaustan takia ketterien kehitysmenetelmien yhteydessä puhutaan useimmiten erilaisten ketterien menetelmien soveltamisesta, on huomattava, että ketterä kehittäminen ei ole vain joukko erilaisia menetelmiä, joilla pyritään tuomaan nopeutta ja joustavuutta suunnitteluun, vaan myös laajempi filosofia tai asenne siihen, miten järjestelmiä tulisi suunnitella ja kehittää. Esimerkiksi Goldmanin et al. (1995) mukaan ketterällä kehityksellä pyritään vastamaan nopeasti niihin haasteisiin, joita nykyaikainen voimakkaasti kilpailtu yritysmaailma tuottaa ja varautumaan jatkuvaan muutokseen. Lisäksi he korostavat sitä, että ketterissä kehitysmenetelmissä asiakkaat ovat keskeisiä ja suunnittelusta pyritään tekemään sellaista, että asiakkaiden tarpeisiin pystytään vastaamaan tehokkaammin ja nämä tarpeet ohjaavat suunnittelua.

Tampereen teknillisen yliopiston kirjastonjohtaja Minna Niemi-Grundström (2012) on todennut, että ketterässä kehityksessä olennaista on konkreettisten menetelmien ohella asenne, jolla kehittämistä tehdään. Hänen mukaansa asiakaslähtöisyys ja toiminnan jatkuva kehittäminen ovat keskeisiä tavoitteita ketterässä kehittämisessä. Asiakaslähtöisyys tarkoittaa tiivistä yhteyttä asiakkaisiin suunnittelun kaikissa vaiheissa ja palautteen keräämistä eri tavoilla. Kehittämistä ei saa nähdä vain laajoina ylhäältä johdettuina projekteina, vaan sen tulisi olla jatkuvaa, joustavaa ja lähteä käytännön tarpeista. Asiakaslähtöisyyden lisäksi suunnittelussa pitäisi korostaa suunnittelijoiden merkitystä ja mahdollistaa heidän jatkuvan oppimisensa ja kehittymisensä. Lisäksi tavoitteena on erilaisten kokeilujen toteutuskynnyksen madaltaminen. (Niemi-Grundström 2012) Myös Karlström ja Runeson (2005, 43) ovat monessa asiassa samoilla linjoilla Niemi-Grundströmin kanssa. Heidän mukaansa ketterien kehitysmenetelmien käytännön soveltamisessa keskeisiä periaatteita ovat pyrkimys yksinkertaisuuteen, suunnittelijoiden korostamiseen käytännön suunnittelutyössä ja keskittyminen järjestelmän kehittämiseen.

2.2 Ketterän kehityksen menetelmät

2.2.1 Kehitysmallit

Ketterä kehitys ei ole vain yksi tietynlainen suunnittelun tapa, vaan se sisältää paljon erilaisia menetelmiä ja ajatustapoja siitä, miten kehittämistä pitäisi toteuttaa. Jo näiden menetelmien ja ajattelutapojen kirjo on hyvin laaja. Lisäksi menetelmiä ja ajattelutapoja voidaan yhdistää suunnittelumalleiksi, joissa sovelletaan tiettyjä lähestymistapoja jonkin suunnittelukokonaisuuden toteuttamiseen. Erilaisia menetelmiä yhdistelemällä ja soveltamalla niitä erilaisissa yhteyksissä, kuten informaatiojärjestelmien suunnittelussa, potentiaalisten suunnittelumallien määrä on valtava.

Ketterän kehityksen menetelmiä voidaan kuitenkin jakaa monin tavoin laajempiin kokonaisuuksiin sen mukaan, miten ne lähestyvät suunnittelua ja millaisia ketterän kehityksen menetelmiä ne soveltavat. Rashmi Jain ja Anithashree Chandrasekaran (2009) ovat jaotelleet ketteriä kehitysmenetelmiä laajempiin kehitysmalleihin ja tarkastelleet näiden hyödyntämismahdollisuuksia ja soveltuvuutta erilaisiin projekteihin. Jain ja Chandrasekaran arvioivat erilaisten ketterän kehityksen menetelmien soveltuvuutta erilaisiin tarkoituksiin sen perusteella, toteuttavatko tai voisivatko menetelmät toteuttaa tiettyjä ominaisuuksia, jotka Jainin ja Chandrasekaran mukaan mittaavat menetelmän ”ketteryyttä”, eli lisäävät kehittämisen nopeutta ja joustavuutta. Näitä tarkasteltuja ominaisuuksia olivat esimerkiksi muokattavuus, kustannusarviointi, teknologisten kompromissien tekeminen, riskien hallinta, menetelmän iteratiivisuus ja testattavuus.

Jainin ja Chandrasekaranin (2009) mukaan ketterät kehitysmenetelmät jakaantuvat neljään pääryhmään: muokattuun vesiputousmalliin (modified waterfall), evolutionaariseen kehitykseen (evolutionary development), kustannuspohjaiseen suunnitteluun (design to cost) ja työkalupohjaiseen suunnitteluun (design to tools). Näistä muokattu vesiputousmalli ja erityisesti evolutionaarinen malli ovat työni kannalta mielenkiintoisia. Perinteinen vesiputousmalli tarkoittaa suunnittelumallia, jossa suunnittelua edistetään vaihe kerrallaan ”putouksittain”, eli yhden vaiheen valmistumisen jälkeen siirrytään seuraavaan vaiheeseen. Perinteinen vesiputousmalli on siis esimerkki lineaarisesta suunnittelusta, jossa yhdestä suunnittelun vaiheesta siirrytään aina yksisuuntaisesti seuraavaan vaiheeseen, eikä aiempiin suunnittelun vaiheisiin

enää palata, kun ne käyty kerran läpi. Muokatussa vesiputousmallissa tähän lineaariseen malliin lisätään iteratiivisuutta[1], palautteen keräämistä ja eri vaiheiden yhtäaikaista suorittamista, joilla saadaan joustavuutta suunnitteluun. Muokattuun vesiputousmalliin perustuvissa menetelmissä voidaan myös jakaa projekti erilaisiin alaprojekteihin (komponentteihin), jotka suoritetaan erikseen. (emt. 32)

Evolutionaarisessa kehityksessä suunnittelun jakaminen viedään pidemmälle ja keskitytään yleensä kerralla vain yhteen järjestelmän ominaisuuteen. Järjestelmän kehittäminen pilkotaan pienempiin osiin ja muokatun vesiputousmallin mukaisesti näiden osien suunnittelu jaetaan vaiheisiin ja korostetaan suunnittelussa iteratiivisuutta. Evolutionaarisissa menetelmissä pyritään yleensä rakentamaan järjestelmää ominaisuus kerrallaan niin, että jokainen osa käy läpi iteratiivisen syklin, jossa se rakennetaan, testataan, integroidaan ja siitä kerätään palautetta. Jainin ja Chandrasekaranin analyysin perusteella erilaiset evolutionaariset kehitysmallit kuuluvat kaikkein ketterimpien joukkoon artikkelissa tarkastelluista menetelmistä, koska niillä on paljon sellaisia ominaisuuksia, jotka nopeuttavat järjestelmän kehittämistä. (emt. 32–33)

Evolutionaaristen kehitysmenetelmien muunnelmista evolutionaarisessa prototyypittämisessä pyritään suunnittelemaan ensin järjestelmän näkyvimmät osat, jolloin järjestelmää voidaan esitellä käyttäjille varhain ja konkreettisesti. Järjestelmää kehitetään koko ajan käyttäjiltä saadun palautteen perusteella. Mielenkiintoinen piirre lähestymistavassa on se, että järjestelmää pidetään mallissa valmiina siinä vaiheessa, kun asiakas on yhtä mieltä suunnittelijoiden kanssa siitä, että järjestelmä on valmis. Tällainen lähestymistapa toimii hyvin tilanteissa, joissa järjestelmän vaatimukset muuttuvat nopeasti ja halutusta lopputuloksesta ei ole vielä tarkkaa käsitystä. Toinen hieman evolutionaarisesta prototyypittämisestä muokattu lähestymistapa on vaiheittainen toimittaminen (staged delivery), jossa toisin kuin prototyypittämisessä ei projektin alussa tarkalleen tiedetä mihin pyritään. Järjestelmää kehitetään iteratiivisesti niin, että siihen luodaan ominaisuus kerrallaan toimintoja ja nämä toiminnot otetaan käyttöön ennen

[1] Iteratiivisuus tarkoittaa järjestelmän suunnittelun yhteydessä johonkin suunnittelun tavoitteeseen pyrkimistä toistamalla samankaltaista suunnittelukokonaisuutta kerta toisensa jälkeen niin, että koko ajan lähestytään haluttua lopputulosta. Iteraatio puolestaan tarkoittaa tässä työssä tällaisen suunnittelukokonaisuuden toistamista yhden kerran. Jonkun suunnittelun kokonaisuuden toteutus koostuu iteratiivisessa suunnittelussa siis useista iteraatioista, joilla suunnittelua parannetaan koko ajan kohti haluttua lopputulosta.

seuraavan ominaisuuden kehittämistä, eikä prototyypittämisen tavoin oteta vasta lopuksi koko järjestelmää käyttöön. (emt. 33)

2.2.2 Prototyypit ja iterointi

Ketterien kehitysmenetelmien soveltamisessa informaatiojärjestelmien suunnitteluun Maryam Alavin artikkeli *An Assessment of the Prototyping Approach to Information Systems Development* (1984) on iästään huolimatta edelleen usein viitattu perusteksti. Alavi vertailee artikkelissa ketterien kehitysmenetelmien soveltamista perinteiseen informaatiojärjestelmien kehittämisen elinkaarimalliin (life cycle approach) ja tarkastelee ketterille kehitysmenetelmille keskeistä prototyyppien rakentamista suunnittelun välineenä. Alavi tutkii prototyyppien vaikutuksia järjestelmien suunnitteluun kahdella tavalla. Haastatteluilla selvitetään prototyyppejä soveltaneiden projektien johtajien ja suunnittelijoiden näkemyksiä prototyyppien käytön hyödyistä ja haasteista. Näitä kvalitatiivisia näkemyksiä täydennetään kontrolloiduilla kokeilla, joissa kahdesta testiryhmästä toinen kehittää informaatiojärjestelmää ketterien kehitysmenetelmien avulla ja toinen elinkaarimallin pohjalta.

Prototyyppien keskeinen hyöty on se, että niiden avulla käyttäjät voivat aidosti ymmärtää järjestelmän toiminnan ja tämän pohjalta arvioimaan ja antamaan palautetta järjestelmän toimivuudesta. Alavin mukaan on yleistä, että yhteys käyttäjiin puuttuu järjestelmien suunnittelussa. Prototyyppien käyttämisen etuna on se, että niiden avulla voidaan konkreettisesti demonstroida ja viestiä käyttäjille, millainen järjestelmä tulee olemaan ennen kuin se otetaan käyttöön. Näin järjestelmää voidaan kehittää vastaamaan käyttäjien tarpeita. Käyttäjät pystyvät Alavin mukaan osoittamaan tehokkaasti olemassa olevien järjestelmien puutteita, mutta ovat huonoja ennustamaan etukäteen tarpeitaan. Koska prototyypit mahdollistavat tehokkaamman palautteen antamisen, ne myös sitouttavat käyttäjiä projektiin antamalla heille mahdollisuuden osallistua järjestelmän kehittämiseen ja vähentävät näin uuden järjestelmän aiheuttamaa muutosvastarintaa. (emt. 557)

Tärkeä huomio projektin hallittavuuden ja suunnittelijoiden oppimisen kannalta on se, että prototyyppien käyttäminen ja kehittämiseen liittyvä kokeileminen eivät auta pelkästään käyttäjiä ymmärtämään järjestelmää. Alavin (1984, 563) mukaan myös järjestelmän suunnittelijat

oppivat prototyyppiä rakentaessaan ja pystyvät vapaammin kokeilemaan erilaisia lähestymistapoja järjestelmän toteuttamiseen ennen kuin se toteutetaan kokonaisuudessaan.

Haasteina prototyyppien käyttämisessä ovat niiden vaatimat resurssit ja se, että prototyyppien hyödyntäminen vaatii erilaista osaamista verrattuna muuhun suunnitteluprojektin toteuttamiseen. Alavin tutkimuksessa myös käyttäjien innostuksen ylläpitäminen koettiin haasteeksi, koska prototyyppiprojekteja pidettiin vain varhaisena ja tilapäisenä versiona järjestelmästä. Toisaalta Alavi toteaa, että mikään ei estä tarvittaessa yhdistämästä ketterien kehitysmenetelmien hyödyntämistä perinteisempään elinkaarimalliin, jolloin erityisesti laajoissa projekteissa elinkaarimallin korostamat valvontamekanismit ja ennalta määritellyt tavoitteet voivat auttaa projektin toteutusta. (emt. 558)

2.2.3 Pilottiprojektit

Ketterien kehitysmenetelmien ohella järjestelmien toteuttamisessa ja erityisesti käyttöönotossa käytetään usein hyväksi pilottiprojekteja, joissa järjestelmiä testataan lopullista tavoitetta pienemmällä käyttäjämäärällä. Tavoitteena pilottiprojekteissa on kerätä kokemuksia järjestelmän käytöstä autenttisemmassa ympäristössä. Pilottiprojekteja voidaan yhdistää ketterillä kehitysmenetelmillä toteutettuihin suunnitteluprojekteihin. Esimerkiksi Jørgen Bansler ja Erling Havn (2010) ovat epäonnistuneen tanskalaisen terveystietojärjestelmän pilottiprojektin toteutusta käsittelevässä artikkelissaan tuoneet esille pilottiprojektien mahdollisuuksia osana joustavampaa järjestelmäsuunnittelua ja ketterien kehitysmenetelmien käyttöä. Bansler ja Havn pohtivat artikkelissa myös sitä, miten käyttäjänäkökulma voidaan huomioida paremmin suunnittelussa.

Banslerin ja Havnin mukaan prototyyppejä käytetään suunnitteluvaiheessa palautteen keräämiseen käyttäjiltä järjestelmän toiminnasta ja ne auttavat käyttäjiä arvioimaan konkreettisesti millaisia vaatimuksia he asettavat järjestelmälle. Prototyyppien käyttämisellä on kuitenkin rajoitteensa. Prototyypit eivät vastaa aidosti käyttöönotettua järjestelmää, koska niitä testataan yleensä laboratorio-oloissa, ei käyttäjien luontaisessa ympäristössä. Näihin prototyyppien puutteisiin voidaan etsiä ratkaisuja pilottiprojekteilla, joiden avulla voidaan prototyyppien tavoin kerätä tietoa järjestelmän toimivuudesta loppukäyttäjiltä ja paljastaa suunnitteluvirheitä ennen lopullisen järjestelmän valmistumista. Verrattuna prototyyppeihin pilottiprojektit toteu-

tetaan kuitenkin aidoissa toimintaympäristöissä ja pilottijärjestelmän käyttöönotto vastaa enemmän todellista järjestelmän käyttöönottoa. Pilottiprojekteissa tavoitteena ei yleensä ole ottaa järjestelmää käyttöön, vaan kerätä kokemusta käyttöönotosta ja hankkia tietoa järjestelmän potentiaalisesta hyödyllisyydestä ja käytettävyydestä. Pilottiprojekteja voidaan käyttää tarpeen mukaan joko täydentämään tai korvaamaan prototyyppejä. (Bansler ja Havn 2010, 638, 646)

2.2.4 Suunnittelun arviointi

Aiemmissa luvuissa esitellyt ketterän kehityksen menetelmät eivät huomioi kovin hyvin joitakin suunnittelun keskeisiä osa-alueita. Erityisesti laatunäkökulman huomioiminen suunnittelussa voi jäädä ketterän kehityksen perusmenetelmillä vajavaiseksi. Ketteriä kehitysmenetelmiä on kuitenkin mahdollista täydentää laatunäkökulmaa korostavilla menetelmillä. Gary Johnston (2005) on pohtinut, miten sähköisen asiakirjajärjestelmän suunnittelussa voisi lisätä suunnittelun laatua hyödyntämällä jatkuvaa arviointia yhdessä ketterien kehitysmenetelmien kanssa. Johnstonin kritisoi nykyisiä asiakirjajärjestelmien suunnittelumalleja, jotka ovat hänen mukaansa liian lineaarisia ja perustuvat edelleen pääosin epärealistiseen vesiputousmalliin ja eivät tämän takia kykene arvioimaan suunnittelun etenemistä kriittisesti. (Johnston 2005)

Suunnittelun edetessä suunnittelun tavoite muuttuu Johnstonin mukaan usein huomaamatta organisaation toiminnan tukemisesta itsetarkoituksselliseksi teknisen järjestelmän suunnitteluksi. Johnston esittelee artikkelissa DIRKS-malliin pohjautuvan uuden mallin, jossa hän on muokannut DIRKS-mallia ketterän kehityksen periaatteita paremmin vastaavan niin sanotun W-mallin pohjalta. W-mallissa jokaisen suunnitteluvaiheen rinnalla suunnittelun tuloksia samalla testataan ja arvioidaan sen sijaan, että odotettaisiin projektin loppuvaiheeseen ennen testaamisen ja arvioinnin aloittamista. Näin saadaan Johnstonin mukaan suunnitteluun joustavuutta ja voidaan paremmin valvoa suunnitellun järjestelmän laatua suunnittelun eri vaiheissa ja varmistaa, että järjestelmä vastaa sille asetettuja vaatimuksia. (emt.)

2.3 Käyttäjien merkitys sähköisen asiakirjajärjestelmän suunnittelussa

Sähköisten asiakirjajärjestelmien suunnittelusta ja käyttöönotosta on kirjoitettu useita tapaustutkimustyyppisiä artikkeleita, joissa käsitellään käytännönläheisesti jotain sähköisen asiakirjajärjestelmän toteutusprojektia (ks. esim. Smyth 2005; Wilkins 2009). Näissä artikkeleissa ei kuitenkaan ole yleensä pohdittu laajemmin sitä, miksi projektit epäonnistuvat ja mitkä tekijät vaikuttavat eniten onnistumiseen. Harvoja kattavia tutkimuksia aiheesta ovat Johanna Gunnlaugsdottirin väitöskirja *The Implementation and Use of ERMS: A Study in Icelandic Organizations* (2006) ja sen pohjalta kirjoitettu artikkeli *As you sow, so you will reap implementing ERMS* (2008). Artikkelissa tarkastellaan, millaiset tekijät vaikuttavat sähköisen asiakirjajärjestelmän (ERMS) käyttöönottoon keskisuurissa organisaatioissa. Gunnlaugsdottir on käyttänyt tutkimuksensa metodina avoimia haastatteluita ja osallistujien tarkkailua.

Gunnlaugsdottir korostaa artikkelissaan toistuvasti käyttäjänäkökulman keskeisyyttä sähköisen asiakirjajärjestelmän suunnittelussa ja myös tutkimuksen tuloksissa käyttäjälähtöiset haasteet todettiin keskeisiksi erottaviksi tekijöiksi onnistuneiden ja epäonnistuneiden projektien välillä. Käyttöönoton onnistumisen kanssa voimakkaimmin korreloineet tekijät olivat johdon aktiivinen ja näkyvä tuki, käyttäjien osallistuminen projektiin ja käyttäjien riittävä koulutus. Korrelaatio näiden tekijöiden ja järjestelmän käyttöaktiivisuuden ja tavoitteiden saavuttamisen välillä oli lähes täydellinen. (Gunnlaugsdottir 2008, 27–28)

Käyttäjien kannalta muutosprojekteissa on Gunnlaugsdottirin mukaan haasteena se, että yksilöt pelkäävät tuntematonta ja informaation jakamisen tärkeys korostuu. Käyttäjien on päästävä osallistumaan päätöksentekoon ja tavoitteiden määrittelyyn, että järjestelmästä pystytään kehittämään käyttäjien tarpeita vastaava. Gunnlaugsdottirin tutkimuksen perusteella käyttäjien aktiivinen osallistuminen projekteihin edesauttoi sitä, että järjestelmästä tehtiin sellainen, jota käyttäjät paitsi pystyivät, myös halusivat käyttää. On tärkeää, että asiakirjajärjestelmäprojektin vaikutusten kohteena olevien osallistumisen on aitoa niin, että heidän mielipiteitään tutkitaan aktiivisesti ja ne myös huomioidaan kehittämisessä. (emt, 23, 32)

Lynette Downing (2006) on puolestaan tutkinut, miten käyttäjänäkökulma pitäisi huomioida dokumenttienhallintajärjestelmien suunnittelussa. Downingin mukaan dokumenttienhallintajärjestelmien käyttöönotossa teknologia on selkeästi vähemmän tärkeää kuin käyttäjiin ja or-

ganisaatiokulttuuriin liittyvät kysymykset. Käyttäjät eivät koskaan käytä dokumenttienhallintajärjestelmää halutulla tavalla vain siksi, että se on olemassa. Heidät on houkuteltava käyttämään järjestelmää. Tässä auttavat esimerkiksi käyttöönottoprosessin tekeminen läpinäkyväksi ja käyttöliittymän suunnittelu niin, että se vastaa käyttäjien tarpeita. Myös käyttäjien odotukset täytyy huomioida. Käyttäjille ei pidä uskotella järjestelmästä liikoja, vaan on tärkeää viestiä tehokkaasti eduista, joita järjestelmä todella tuo heille – erityisesti henkilökohtaisesti.

2.4 Informaatiojärjestelmän suunnittelu oppimisprosessina

Vaikka suunnittelijoiden oppiminen suunnittelun aikana tuntuu tärkeältä näkökulmalta järjestelmien suunnittelun onnistumisen kannalta, sitä on kirjallisuudessa käsitelty vain vähän ja harvoissa löytämissäni asiaa käsittelevissä artikkeleissakin se on lähinnä vain sivujuonne muiden asioiden käsittelyn yhteydessä. Joanne Evans ja Nadav Rouche (2006) ovat kuitenkin pohtineet ketterien kehitysmenetelmien ja suunnittelijoiden oppimisen suhdetta. Heidän mukaansa järjestelmän suunnittelussa on usein järkevää soveltaa erilaisia käyttäjäkeskeisiä ketterän kehityksen menetelmiä, joissa korostuu lyhyt suunnittelun, käyttöönoton ja arvioinnin sykli, jolla pyritään iteratiivisesti hankkimaan tietoa järjestelmän vaatimuksista. Evansin ja Rouchen mielestä tällaisten ketterien kehitysmenetelmien soveltaminen edistää suunnittelijoiden oppimista, koska informaatiojärjestelmän suunnittelun voi nähdä – toisin kuin perinteisesti on ajateltu – myös tutkimus- ja oppimisprosessina, jossa ketterien kehitysmenetelmien tuella etsitään järkevimpiä tapoja toteuttaa järjestelmä. Esimerkiksi suunnittelun iteratiivisuus ja prototyyppien rakentaminen mahdollistavat erilaisten määritysten järkevyyden arvioinnin ja syvemmän ymmärryksen luomisen suunnittelun vaatimuksista ja teknologian mahdollisuuksista toteuttaa näitä vaatimuksia. (Nadav ja Rouche 2006, 320–334)

Pilottiprojekteja käsittelevän tutkimuksensa yhteydessä myös Bansler ja Havn ovat pohtineet oppimisen merkitystä suunnittelussa. He korostavat, että epävarmuus ja monitulkintaisuus kuuluvat kiinteästi informaatiojärjestelmien suunnitteluun. Tämän takia tiukka suunnittelu ja projektien kontrollointi eivät toimi suunnittelun pohjana, vaan tarvitaan joustavia, iteratiivisia ja vaiheittaisia lähestymistapoja, jotka korostavat jatkuvaa oppimista ja rohkaisevat kokeilemiseen ja improvisointiin. Järjestelmäkehitys on tekijöille oppimisprosessi. Oppiminen puolestaan vaatii, että pystytään tehokkaasti keräämään palautetta käyttäjiltä ja kehittämään suun-

nittelua suunnitteluvaiheiden välillä. Käyttäjien palautteen kerääminen ei kuitenkaan ole yksinkertaista, koska käyttäjät eivät ymmärrä abstrakteja järjestelmämäärityksiä, eivätkä osaa ennustaa miten järjestelmä vaikuttaa heidän työskentelyynsä. Näihin suunnittelun haasteisiin voidaan Banslerin ja Havnin mukaan vastata parhaiten iterointia tukevilla ketterillä suunnittelumenetelmillä, kuten prototyypeillä ja pilottiprojekteilla. Prototyyppien ja pilottiprojektien hyödyntäminen tukee myös käyttäjien osallistumista suunnitteluun. (Bansler ja Havn 2010, 637–638)

Jim Highsmith (2009) on puolestaan kirjassaan *Agile Project Management: Creating Innovative Projects* pohtinut ketterien kehitysmenetelmien soveltamista projektienhallintaan ja projektin toteuttajien johtamiseen. Highsmithin (2004, 255) mukaan joustava suunnittelu vaatii kokeilullisuuteen kannustamista ja muutosta perinteisiin asenteisiin, jotka korostavat tiukkaa suunnitelmallisuutta. Kokeiluun kannustaminen on tärkeää, koska sen kautta tapahtuva oppiminen on tehokas keino testata mitkä lähestymistavat toimivat ja mitkä eivät ja kehittää uusia innovaatioita. Ketterä projektinhallinta ja työntekijöiden oppimisen kannustaminen luovat mukautuvia ryhmiä, jotka pystyvät paremmin täyttämään asetetut tavoitteet joustavasti. (Highsmith 2004)

3 TUTKIMUSASETELMA

3.1 Tutkimusongelma

Tutkin työssä, kuinka sähköisten asiakirjajärjestelmien tietosisällön suunnittelua voidaan nopeuttaa ja tehdä joustavammaksi ketterien kehitysmenetelmien avulla ja miten tällainen ketterämpi suunnittelu vaikuttaa suunnittelun hallittavuuteen kokemattoman suunnittelijaryhmän näkökulmasta.

Tutkimus vastaa seuraaviin kysymyksiin:

1. Miten lineaarista sähköisen asiakirjahallintajärjestelmän tietosisällön suunnittelua voi ketterien kehitysmenetelmien avulla nopeuttaa ja tehdä joustavammaksi?
2. Kuinka ketterien kehitysmenetelmien soveltamiseen pohjautuva suunnittelumalli vaikuttaa kokemattomien suunnittelijoiden kykyyn hallita suunnitteluprojektia?

Työn hypoteesi on, että ketteriä kehitysmenetelmiä sähköisen asiakirjajärjestelmän suunnitteluun lisäämällä voidaan lisätä suunnittelun joustavuutta ja tulosten saavuttamisen nopeutta, ja että tämä lisää suunnittelun hallittavuutta kokemattomalle suunnittelijaryhmälle.

Sähköisellä asiakirjajärjestelmällä (electronic records management system tai ERMS) tarkoitetaan työssä ohjelmistoa, joka on suunniteltu erityisesti asiakirjojen hallintaan. Asiakirjoja käsitellään organisaatioissa myös niin sanotuissa liiketoimintajärjestelmissä, joita organisaation työntekijät käyttävät työtehtäviensä suorittamiseen. Liiketoimintajärjestelmiä voivat olla esimerkiksi tietokantojenhallintaohjelmat tai verkkosisällön hallintaan käytetyt ohjelmistot. (National Archives of Australia 2001, A User's Guide, 12) Tässä työssä tarkastelen vain sähköisten asiakirjajärjestelmien suunnittelua ja rajaan asiakirjojen käsittelyyn käytetyt liiketoimintajärjestelmät pois tarkastelusta.

Asiakirjahallinnan yhteydessä käytetään usein myös termiä asiakirjajärjestelmä (records system). Asiakirjajärjestelmä on ISO 15489-1 -standardin suomenkielisen version mukaan ”tietojärjestelmä, joka ottaa talteen ja käsittelee asiakirjoja sekä mahdollistaa pääsyn niihin elinkaaren kaikissa vaiheissa”. Määrittelyn termi ”tietojärjestelmä” tarkoittaa tässä yhteydessä paitsi ohjelmistoja, joissa asiakirjoja käsitellään, myös muita asiakirjojen hallintaan vaikuttavia teki-

jöitä, kuten henkilöstöä ja erilaisia toimintaperiaatteiden ja standardien kaltaisia asiakirjahallinnan ohjausvälineitä. (Suomen standardisoimisliitto 2007, 9, 16–17) Samoin DIRKS-oppaan määritelmän mukaan asiakirjajärjestelmään luetaan paitsi asiakirjojen tuottamiseen ja hallintaan käytettävät tietojärjestelmät, myös asiakirjojen käyttäjät ja asiakirjahallinnan ammattilaiset; käytänteet; vastuunjaot; ohjeistukset; muut välineet, joilla asiakirjahallintaa ohjataan; asiakirjahallintaan käytetyt laitteistot; asiakirjahallinnan ohjelmistot; muut välineet sekä itse asiakirjat (National Archives of Australia 2001, 12). Asiakirjajärjestelmä on siis laaja näkemys siitä, mikä vaikuttaa asiakirjojen hallintaan, eikä rajoitu vain ohjelmistoihin, joita käytetään hallinnan apuna.

Tässä työssä näkökulmani sähköisen asiakirjajärjestelmän suunnitteluun ei ole niin laaja kuin ISO-standardissa ja DIRKS-oppaassa. Käsittelen vain asiakirjojen hallintaan tarkoitettujen ohjelmistojen tietosisällön suunnittelua. Tietosisällöllä tarkoitan sähköiseen asiakirjajärjestelmään laadittavia asiakirjahallinnan välineitä, kuten tehtäväluokitusta, metatietoja, prosessikuvauksia ja muita sisältöjä, joiden avulla asiakirjoja hallitaan järjestelmässä ja joiden suunnittelussa asiakirja-ammattilaisilla on keskeinen vastuu. Suunnittelulla tarkoitan kaikkia rajaukseeni kuuluvia sähköisen asiakirjajärjestelmän tietosisällön tuottamiseen liittyviä toimenpiteitä. Tällaisia ovat esimerkiksi tietosisällön konkreettinen tuottaminen ja tietosisällön testaamiseen käytettävien välineiden suunnittelu.

En käsittele työssä sähköisen asiakirjajärjestelmän teknistä toteutusta, kuten ohjelmiston suunnittelua tai asentamista. En myöskään käsittele DIRKS-oppaassa esiteltyä asiakirjajärjestelmän kokonaisvaltaisempaan suunnitteluun kuuluvaa kokonaisuutta, johon sisältyvät esimerkiksi asiakirjahallinnan strategioiden valinta, vanhojen järjestelmien analysointi ja muiden kuin sähköisen asiakirjajärjestelmän hyödyntäminen asiakirjojen hallinnassa. Tarkoituksena on pitää malli konkreettisena ja kuvata niitä vaiheita, joissa järjestelmän tietosisältöä suunnitellaan ja ottaa laajempia näkökulmia mukaan vain siltä osin kun se on tietosisällön suunnittelun kannalta tarpeellista.

Aiempaa tutkimusta käsittelevässä luvussa esittelin erilaisia näkemyksiä siitä, mikä tekee menetelmistä ketteriä. Tässä työssä tarkoitan ketterällä kehittämisellä suunnittelua, joka:

- nopeuttaa tulosten saavuttamista
- lisää suunnittelun joustavuutta

Tuloksilla en tarkoita pelkästään järjestelmän valmistumista aktiivikäyttöön, vaan myös erilaisten välivaiheiden ja välitavoitteiden saavuttamista, kuten jonkun ennalta määrätyn kokonaisuuden suorittamista valmiiksi niin, että voidaan siirtyä seuraavaan suunnitteluvaiheeseen tai aloittaa valmiiksi suoritetun vaiheen suunnittelu alusta entisten tulosten päälle. Joustavuudella tarkoitan sitä, miten suunnittelumalli sallii liikkumisen eri suunnitteluvaiheiden välillä myös muuten kuin lineaarisesti seuraavaan vaiheeseen. Lisäksi arvioin toteutuksen joustavuutta sen pohjalta lisääkö se suunnittelun käyttäjälähtöisyyttä, mikä tarkoittaa käytännössä useimmiten palautteen keräämistä käyttäjiltä.

Käyttäjien korostaminen joustavuuden kriteerinä pohjautuu omiin kokemuksiini käyttäjien keskeisyydestä suunnittelussa ja kirjallisuudessa esitettyihin esimerkkeihin käyttäjien keskeisyydestä suunnittelun onnistumisessa. Käyttäjänäkökulman korostaminen projekteissa auttaa tutkimusten mukaan tasapainon löytämistä suunnittelun joustavuuden ja projektin edistämisen välillä sekä edistää merkittävästi projektien toteuttamista eri osapuolia tyydyttävällä tavalla ja varmistaa osaltaan järjestelmän laadun toteutumista. (ks. esim. Kautz 2011, 231; Gunnlaugsdottir 2008, 23) Käyttäjänäkökulma on mielenkiintoinen myös siksi, että esimerkiksi Snyderin ja Coxin (1985, 64) mukaan aidosti lineaarisissa suunnittelumalleissa ei ole edes käytännössä mahdollista kerätä riittävästi käyttäjäpalautetta projektin edetessä. Suunnittelun hallittavuuden käsittelyssä pohdin sitä, miten käyttäjälähtöisyys vaikuttaa hallittavuuteen ja kuinka ketteriin kehitysmenetelmiin pohjautuva malli vertautuu lineaarisiin malleihin käyttäjien huomioimisessa.

Suunnittelun hallittavuudella tarkoitan työssä seuraavia asioita:

- kuinka hallittaviin kokonaisuuksiin malli jakaa suunnittelua
- miten malli vaikuttaa suunnittelijoiden mahdollisuuksiin kokeilla erilaisia lähestymistapoja suunnitteluun
- kuinka hyvin malli mahdollistaa suunnittelijoiden oppimisen, eli tiedon ja osaamisen hankkimisen suunnittelun vaatimuksista ja toteuttamisesta suunnittelun aikana

Tarkastelen hallittavuutta erityisesti suunnittelijoiden näkökulmasta ja jätän muiden vaikutusten, kuten hallittavuuden lisääntymisen vaikutukset muutosjohtamiseen, pois. Lisäksi koska käyttäjälähtöisyys on joustavuuden kriteerinä mallin rakentamisessa keskeinen, huomioin sen korostetusti myös suunnittelun hallittavuuden arvioinnissa.

Pohdin työssä mallin pohjalta, onko oletettavaa, että suunnittelu muuttuu hallittavammaksi myös sähköisen asiakirjajärjestelmän suunnittelussa ketterien kehitysmenetelmien avulla vai onko tällaisten järjestelmien suunnittelussa jotain erityispiirteitä, mikä tekee hallittavuuden lisäämisen vaikeaksi ja lineaarisen lähestymistavan järkevämmäksi. Tarkasteltavaksi näkökulmaksi olisi voinut valita perustellusti useita muita keskeisiä ja yleisiä ongelmia asiakirjajärjestelmien suunnittelussa, joihin ketterien kehitysmenetelmien soveltaminen olisi voinut auttaa, kuten Gunnlaugsdottirin (2008) tutkimuksessa esille tulleen johtamisen vaikutuksen tai käyttäjien kouluttamisen ja osallistamisen. Työn rajaamisen takia vain yhden näkökulman mukaan ottaminen on kuitenkin perusteltua.

3.2 Tutkimusmenetelmät

Tarkastelen ketterien kehitysmenetelmien soveltamista sähköisen asiakirjajärjestelmän tietosisällön suunnittelussa kehittämällä lineaarisia asiakirjahallinnan ohjeistuksia pohjana käyttäen uudenlaisen suunnittelumallin, johon lisään joustavuutta ja nopeutta ketterillä kehitysmenetelmillä. Käytän mallin rakentamisen ja sen vaikutusten arvioinnin pohjana Evansin ja Rouchen (2006) informaatiojärjestelmän kehittämisen mallia, joka on suunniteltu tutkimuskäyttöön tarkoitetun informaatiojärjestelmän toteuttamiseen. Evans ja Rouche ovat soveltaneet mallissaan niin sanottuja järjestelmäkehityksen tutkimusmenetelmiä (systems development research methods) tutkimuskäyttöön tarkoitetun metatietorekisterin suunnitteluun.

Evansin ja Rouchen mukaan tutkimuskäyttöön tarkoitetun järjestelmän suunnittelussa on kolme iteratiivista vaihetta:

1. konseptin kehittäminen
2. järjestelmän rakentaminen
3. järjestelmän arviointi

Konseptin rakentamisvaiheessa asetetaan tutkimuskysymys, jota järjestelmällä pyritään ratkaisemaan, ja määritellään toiminnallisuus- ja muut vaatimukset järjestelmälle. Tavoitteena on määritellä, millainen järjestelmästä tehdään ja mitä järjestelmäkehityksen ongelmaa järjestelmän luomisella pyritään ratkaisemaan. Konseptin rakentamisvaiheessa keskeistä on muiden tieteenalojen kirjallisuudesta löytyvien ideoiden hyödyntäminen järjestelmän suunnittelussa.

Konseptin rakentaminen on tutkimuskäyttöön tarkoitetun järjestelmän kehittämisessä keskeistä, koska järjestelmän tavoitteena on testata jonkinlaista järjestelmämallia. Järjestelmän rakentamisvaiheessa luodaan konkreettisesti toimiva järjestelmä, jota voidaan käyttää ja testata. Viimeisessä järjestelmän arvioimisvaiheessa tarkkaillaan järjestelmän toimintaa esimerkiksi laboratorio- tai kenttätutkimuksella ja saatujen kokemusten ja arviointien pohjalta luodaan uusia teorioita siitä, miten järjestelmää voitaisiin kehittää. (Evans & Rouche 2006, 318–320)

Oman mallini kehittelyssä en suorita konkreettista järjestelmän rakentamisvaihetta, joten malli ei ole suoraan sovellettavissa, mutta perusrunko antaa lähtökohdan mallin kehittämiselle. Konseptin kehittäminen rakentuu Evansin ja Rouchen mallissa tutkimusongelman määrittelylle, jonka pohjalta luodaan järjestelmä ja testataan, miten toteutettu malli vaikuttaa johonkin ennalta määriteltyyn tekijään. Tässä työssä rakennan ensin suunnittelumallin niin, että sovellan ketteriä kehitysmenetelmiä lineaarisempiin sähköisen asiakirjajärjestelmän suunnittelumalleihin. Evansin ja Rouchen mallissa järjestelmän testaaminen tapahtuu käytännössä rakentamalla testijärjestelmä, jonka avulla voidaan testata, miten järjestelmä toteutustapa vaikuttaa johonkin ennalta määriteltyyn tekijään. Omassa työssäni tämä tutkittava tekijä on mallin vaikutus projektin suunnittelijoiden kykyyn hallita suunnitteluprojektia. Työn suppeudesta johtuen ei ole realistista, että toteuttaisin järjestelmän käytännössä ja testaisin sitä, vaan korvaan työssä analyysiosan kirjallisuuden ja omien kokemusten pohjalta tapahtuvalla analyysilla.

Mallin rakentamisen menetelmäni muistuttaa osittain myös Johnstonin (2004, 9–12) asiakirjajärjestelmän kehittämisen tarkasteluun käyttämää menetelmää. Johnston hyödyntää niin sanottua järjestelmän kehityksen tutkimusmetodia, jossa pyritään luomaan uudenlainen suunnittelumalli järjestelmän suunnitteluun ja toteuttamiseen. Evansin ja Rouchen tavoin Johnstonin mallissa aloitetaan konseptin rakentamisesta, mikä tarkoittaa tutkimuskysymysten muotoilua ulkoisten lähteiden ja kriittisen ajattelun avulla. Tämän jälkeen rakennetaan itse malli, missä Johnston soveltaa niin sanottua aukkomallia (gap model). Aukkomallissa pyritään ensin tunnistamaan puutteita ja vahvuuksia aiemmissa järjestelmiensuunnittelumalleissa. Tämän jälkeen uuteen malliin lisätään vanhojen mallien vahvuudet ja korjataan havaittuja puutteita. Mallin kehittämisen jälkeen kerätään palautetta luodusta mallista ja palataan takaisin järjestelmän rakentamiseen saadun palautteen pohjalta. Tätä jatketaan niin monta kertaa, että saavutetaan haluttu taso.

Omasta lähestymistavastani Johnstonin malli eroaa erityisesti siinä, että sen lähtökohtana on vanhojen mallien analysointi ja niiden aukkojen paikkaaminen. Evansin ja Rouchen malliin pohjautuvassa menetelmässäni pyrin vastaamaan asettamiini kysymyksiin luomalla tietyillä kriteereillä (nopeus ja joustavuus) mallin ja sen jälkeen tutkin, miten se vaikuttaa tutkittavaan ilmiöön (suunnittelun hallittavuus kokemattomalla suunnitteluryhmällä).

3.3 Ketterän kehityksen menetelmien soveltaminen

Mallin rakentamisessa pyrin rajaamaan tarkastelua soveltamalla vain keskeisimpiä informaatiojärjestelmien suunnittelussa käytettyjä ketteriä kehitysmenetelmiä. Aiemmin tutkimuksen käsittelyn yhteydessä toin esille, miten ketterät kehitysmenetelmät ovat laaja joukko erilaisia lähestymistapoja suunnitteluun ja niitä voidaan yhdistellä lukuisin eri tavoin laajemmiksi suunnittelumalleiksi. Tämän laajuuden takia tarkastelua täytyy rajata vain joihinkin keskeisiin ketterän kehityksen menetelmiin. Koska Jainin ja Chandrasekaranin (2009, 32–33) mukaan evolutionaarinen kehitysmalli on eri malleista ketterimpiä, hyödynnän tarkastelun pohjana sen korostamia ketterän kehityksen menetelmiä:

- iterointia
- prototyypittämistä
- osittamista

Lisäksi korostaakseni mallin rakentamisessa käyttäjänäkökulmaa, tarkastelen kahta ketterien kehitysmenetelmien kanssa usein sovellettua ja niiden kanssa hyvin yhteensopivaa suunnittelun testausmenetelmää (Johnston 2004, Bansler & Havn 2010):

- pilottiprojekteja
- jatkuvaa arviointia

Iteroinnin ohella evolutionaarisessa mallissa prototyyppien hyödyntäminen on keskeistä. Prototyyppejä käytetään usein ketterän kehityksen menetelmien osana, koska iteraatioiden tavoitteena ei lineaaristen mallien tavoin yleensä ole lopullinen järjestelmä, vaan lähempänä tavoiteltua lopputulosta oleva välivaihe. Välivaiheen tulosten testaaminen vaatii konkreettisia välineitä ja tähän käytetään usein prototyyppejä, joilla voidaan havainnollistaa saavutettuja tuloksia. Sähköisten asiakirjajärjestelmien tietosisällön suunnittelussa prototyypit voivat olla esi-

merkiksi erilaisia kokeilujärjestelmiä, joissa senhetkistä suunnittelun tilannetta voi kokeilla käytännössä.

Ketterille kehitysmenetelmille tyypillistä on myös projektien osittaminen, eli jakaminen pienempiin ja hallittavampiin kokonaisuuksiin. Myös evolutionaarisessa kehittämismallissa käytetään osittamista ja hyödynnän sen osittamismallia tarkastelun pohjana. Osittamiseen liittyen evolutionaarinen lähestymistapa hyödyntää työvaiheiden rinnakkaista suorittamista. Rinnakkainen suorittaminen tarkoittaa sitä, että suunnittelukokonaisuuden eri osia suoritetaan samaan aikaan sen sijaan, että seuraava suunnittelukokonaisuuden osa aloitettaisiin vasta sitten, kun edellinen on kokonaan valmis. Rinnakkaista suorittamista ei kirjallisuuden perusteella hyödynnetä informaatiojärjestelmien suunnittelussa niin paljoa kuin ohjelmistoprojekteissa ja en tämän takia käsittele sitä omana kokonaisuutenaan, mutta hyödynnän kuitenkin sitä, jos se on luontevaa.

Pilottiprojekteja käytetään prototyyppien tavoin järjestelmän testaamiseen ja palautteen keräämiseen. Pilottiprojektit ovat kuitenkin yleensä prototyyppien rakentamista ja testaamista laajempia kokonaisuuksia, joissa jotakin suunnittelun kokonaisuutta testataan käyttäjien aidossa toimintaympäristössä huolellisesti suunnitellulla testijärjestelmällä. Pilottiprojektit eivät tiukasti tulkittuna ole ketterän kehityksen menetelmiä, mutta niitä hyödynnetään usein niiden yhteydessä, kun halutaan laajemmin tietoa järjestelmän suunnittelusta. En käsittele työssä tarkemmin pilottiprojektien toteuttamista, mutta pohdin niiden käyttömahdollisuuksia erilaajuisten suunnittelukokonaisuuksien testaamisessa.

Jatkuva arviointi on puolestaan erityisesti laadunhallinnan väline, sillä sen avulla pyritään arvioimaan suunnittelun eri vaiheissa sitä, miten suunnittelu vastaa sille asetettuihin tavoitteisiin ja käyttäjien tarpeisiin. Jatkuvassa arvioinnissa korostetaan käyttäjien antamaa palautetta ja luovutetaan heille konkreettisesti valtaa suunnittelupäätöksiin. Ajatuksena on, että arvioinneissa tapahtuvan palautteen pitää todella vaikuttaa suunnittelupäätöksiin. (Johnston 2005)

3.4 Mallin rakentaminen

Aloitan suunnittelumallin rakentamisen luomalla mallin pohjaksi rungon, eli tärkeimmät suunnitteluvaiheet, joita sähköisen asiakirjajärjestelmän tietosisällön suunnittelu vaatii. Rungon rakentamisessa karsin suunnittelun vaiheita niin, että runko pysyy tiiviinä ja hallittavana. Otan mukaan vain sellaiset vaiheet, jotka ovat tietosisällön tuottamisen kannalta keskeisiä ja ketterien kehitysmenetelmien soveltamisen kannalta mielekkäitä. Yksittäisten suunnitteluvaiheiden mukaan ottamisella pyrin konkretisoimaan tarkastelua, koska muuten ketteryyden lisääminen jää kovin pinnalliseksi ja etäiseksi konkreettisista suunnitteluvalinnoista.

Mallin rungon rakentamisen jälkeen sovellan erilaisia ketterän kehityksen menetelmiä mallin rungon päälle. Sovellan ketteriä kehitysmenetelmiä koko tietosisällön suunnittelun, pienempien osakokonaisuuksien ja yksittäisten suunnittelun vaiheiden tasolla. Ketterän kehityksen menetelmien soveltamisen malliin toteutan etsimällä ensin kirjallisuudesta esimerkkejä siitä, miten menetelmiä voi soveltaa suunnitteluun. Tämän jälkeen lisään näitä sovellustapoja malliini, jos ne lisäävät tietosisällön suunnittelun nopeutta ja joustavuutta. Jos nopeuden ja joustavuuden tavoitteet ovat keskenään ristiriidassa tai näiden kriteerien pohjalta ei ole muuten mielekästä arvioida ketterän kehityksen menetelmän soveltamisen toimivuutta, arvioin loogisesti ja omien kokemusteni pohjalta, toimiiko lähestymistapa tietosisällön suunnittelussa.

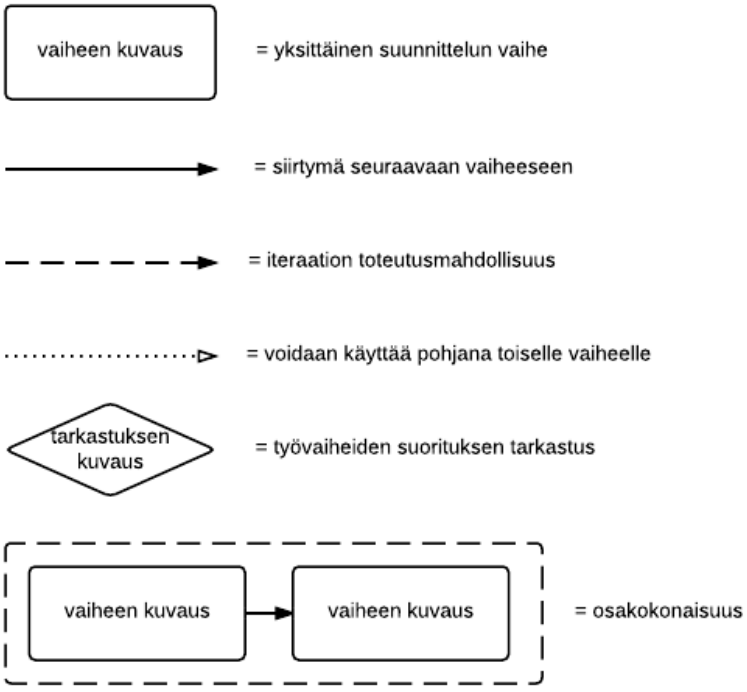
Käytännössä rakennan mallin niin, että otan tarkasteluun yksi kerrallaan jonkin ketterän kehityksen menetelmän ja käsittelen tämän menetelmän soveltamista sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun irrallaan muiden menetelmien soveltamisesta. Sovellan yksittäisiä menetelmiä siis pääosin pelkän mallin rungon päälle. En esimerkiksi ota iteroinnin käsittelyssä malliin jo pilottiprojektien käsittelyssä lisättyjä osioita. Olen valinnut tämän toteutustavan toisaalta siksi, että se pitää käsittelykokonaisuudet selkeämpinä ja toisaalta siksi, että kaikkien osien vaikutusta toisiinsa on vaikeaa arvioida ennen kuin eri menetelmien sovellusmahdollisuudet ovat laajemmin selvillä.

Teen erillisen käsittelyn pääsääntöön kuitenkin jonkin verran poikkeuksia silloin, jos käsiteltävää menetelmää on mielekkäämpää soveltaa jo aiemmin esitellyn menetelmän sovellustavan päälle. Esimerkiksi osittamisen käsittelyssä valitut osittamisen tavat vaikuttavat siihen, miten iterointia on järkevää soveltaa tai millaisille kokonaisuuksille pilottiprojekteja voidaan toteut-

taa. Tällaisissa tapauksissa pohdin myös eri menetelmien yhteyksiä ja piirrän aiemmin hahmotetut menetelmien sovellukset näkyviin mallia esittäviin kuvioihin.

Koska en rakenna mallia luvuissa toistensa päälle, yksittäisten menetelmien sovelluslukujen jälkeen yhdistän lopuksi eri menetelmät kokonaismalliksi omassa luvussaan. Kokonaismallin rakentamisessa sovitan yksittäisiä menetelmiä käsittelevissä luvuissa esitellyt menetelmien sovellustavat yhteen ja luon yhden kokonaismallin, jossa näkyvät eri menetelmien vaikutukset toisiinsa.

Hahmotan mallin rakentamista ja erilaisten menetelmien soveltamista malliin esittämällä aina jokaisen sovellustavan yhteydessä kuvion siitä, miten sovellus liitetään malliin. Käytän kuvioiden piirtämisessä seuraavia elementtejä:



Kuvio 1. Suunnittelumallin kuvauksissa käytettävät elementit

Laatikot kuvaavat yksittäisiä vaiheita, kuten iteraation vaatimusmäärittelyä tai tehtäväluokituksen laatimista. Yhtenäisellä viivalla piirretyt nuolet yhdistävät vaiheita, jotka tietosisällön suunnittelussa on aina suoritettava. Katkoviivanuolet puolestaan kuvaavat sellaisia suunnittelun kohtia, joista voidaan suorittaa iteraatio. Katkoviivanuolia käytetään kuvaamaan myös prototyyppien tai pilottiprojektien käyttämisen mahdollisuutta. Prototyypeissä ja pilottiprojekteissa on käytetty iteroinnin kanssa samaa symbolia siksi, että prototyypit ja pilottiprojektit

sisältävät mallissa aina myös iteroinnin mahdollisuuden. Pisteistä muodostuvat nuoliviivat kuvaavat mallissa sitä, miten joitain suunnittelun osia voidaan käyttää hyödyksi myöhemmässä suunnittelussa. Käytännössä tämä tarkoittaa prototyyppien hyödyntämistä toisten prototyyppien luomisessa. Salmiakkikuvio kuvaa työssä tarkastuspistettä, jossa tarkastetaan ovatko kaikki suunnittelun osat toteutettu niin, että suunnittelussa voidaan edetä seuraavaan vaiheeseen. Katkoviivan sisään kuvatut suunnitteluvaiheet puolestaan kuvaavat jotain suunnittelun osakokonaisuutta, joka on syntynyt suunnittelun osittamisen seurauksena. Tällaisia osakokonaisuuksia käytetään työssä kuvaamaan sitä, miten suunnittelua voi jakaa osiin ja kohdistaa erilaisia ketterän kehityksen menetelmiä myös näihin osakokonaisuuksiin koko suunnittelun ja yksittäisten vaiheiden ohella.

3.5 Kirjallisuuden hyödyntäminen tutkimusprosessissa

Tutkimuksen toteuttamisessa tarvitsen lähteitä neljään eri tarkoitukseen:

- mallin rungon rakentamiseen
- valittujen ketterän kehityksen menetelmien soveltamistapojen kartoittamiseen
- menetelmien soveltuvuuden arviointiin asettamieni kriteerien pohjalta
- mallin suunnittelun hallittavuuteen aiheuttamien vaikutusten arviointiin

Käytän mallin rungon rakentamisessa DIRKS-ohjeistusta (National Archives of Australia 2001) ja arkistolaitoksen AMS-opasta (Kansallisarkisto 2012). Valitsin DIRKS:n ja AMS-oppaan rungon pohjaksi siksi, että ne täydentävät hyvin toisiaan. DIRKS on yleisemmän tason kuvaus asiakirjajärjestelmän suunnittelusta ja suunnitteluprojektin eri vaiheista ja AMS-opas käsittelee yksityiskohtaisemmin tietosisällön suunnittelua.

DIRKS-ohjeistus on paitsi yksi harvoja laajemmin asiakirjahallinnan suunnittelua käsitteleviä ohjeistuksia, se myös kuvaa selkeästi sitä, millaisista vaiheista asiakirjahallinnan suunnittelu koostuu. Tämän pohjalta pystyn hahmottamaan paremmin sitä, millaisia vaiheita suunnittelussa yleisesti käydään läpi ja asettaa niiden pohjalta yleisiä suunnittelun vaiheita ja rajoja myös tietosisällön suunnittelulle. DIRKS-ohjeistus toimii rungon rakentamisessa mallina siitä, millaisia perusvaiheita asiakirjajärjestelmän suunnitteluprojektiin sisältyy. DIRKS:n lisäetuna on

myös se, että siinä on jossain määrin sovellettu myös ketterän kehityksen menetelmiä ja näitä sovelluksia voi käyttää mallina myös eri menetelmien soveltamista käsittelevissä luvuissa.

AMS-oppaan olen valinnut rungon rakentamiseen, koska DIRKS:n suunnittelun ohjeistus on niin yleisluontoinen, että siinä ei käsitellä asiakirjajärjestelmän suunnittelun yksityiskohtia, kuten työssä käsittelemääni tietosisällön tuottamista. DIRKS pyrkii vain kuvaamaan laajemmat suunnittelun vaiheet, joita tarvitaan asiakirjajärjestelmän kokonaisvaltaisessa suunnittelussa. AMS-opas kuvaa asiakirjahallinnan suunnittelusta huomattavasti DIRKS:iä kapeamman osa-alueen. Se kuvaa pääosin sitä, millaista tietosisältöä Suomessa asiakirjahallinnan välineenä laajasti käytettyihin arkistonmuodostussuunnitelmiin pitää suunnitella. Arkistonmuodostussuunnitelmien tietosisältöä voidaan käyttää myös sähköisen asiakirjajärjestelmän tietosisällön pohjana ja siksi AMS-oppaan kuvaama tietosisällön tuottaminen sopii myös tämän työn näkökulmaan. AMS-oppaan pohjalta kuvaan siis mallin runkoon varsinaisen tietosisällön tuottamisen vaiheet DIRKS:n yleisten vaiheiden sisälle.

AMS-opas jakaantuu kahteen osaan: paperiasiakirjoihin suunnitellun AMS:n ja sähköisten asiakirjojen hallinnan erityispiirteet huomioivan AMS:n toteuttamiseen. Vaikka työssä tutkin sähköisten asiakirjajärjestelmien suunnittelua, hyödynnän lähinnä AMS-oppaan paperiasiakirjojen käsittelyä sisältävää osaa. Tietosisällön suunnittelun perusvaiheet on kuvattu tarkemmin paperiasiakirjoja koskevassa osiossa ja se soveltuu siksi paremmin mallin rungon suunnitteluun. AMS-oppaan sähköisen asiakirjajärjestelmän toteutusta kuvaavassa osassa lähinnä täydennetään paperiasiakirjoja koskevaa osuutta. Nämä täydennykset, kuten rekisteröinti, käsittelyprosessien ohjaaminen tai versioiden hallinta, eivät ole tarpeellisia mallin rungon rakentamisessa, koska ne käsittelevät lähinnä sähköisen asiakirjajärjestelmän toiminnallisuuden ja asiakirjojen hallinnan toteuttamista, eivät tietosisällön tuottamista.

Kirjallisuuden hyödyntäminen suunnittelumallin rakentamisessa jakaantuu luvun alussa esittämäni jaottelun mukaisesti kahteen osaan. Ensin etsin valitsemastani kirjallisuudesta erilaisia tapoja soveltaa ketteriä kehitysmenetelmiä ja mikäli tällaisen lähestymistavan soveltaminen tuntuu mahdolliselta ja loogisesti mielekkäältä, sijoitan sen malliin. Tämän jälkeen arvioin kirjallisuuden tuella esitetyn sovelluksen järkevyyttä ja vaikutuksia suunnittelun nopeuteen ja joustavuuteen. Kirjallisuutta käytetään tarkastelussa siis kahdessa roolissa: menetelmien soveltamistapojen tunnistamisessa ja soveltamisen vaikutusten arvioinnissa. Osaa kirjallisuudesta käytän molemmissa rooleissa, koska ne käsittelevät sekä menetelmien soveltamista, että

arvioivat niitä. Soveltamisen arviointi pohjautuu kirjallisuuden lisäksi myös omiin kokemuksiini sähköisten asiakirjajärjestelmien suunnittelusta ja käyttämisestä.

Myös mallin rakentamisen jälkeen seuraava mallin vaikutusten arviointi suunnittelun hallittavuuteen pohjautuu kirjallisuuden hyödyntämiseen ja omiin kokemuksiini. Erona mallin rakentamisessa käytettyyn kirjallisuuteen on se, että hyödynnän kirjallisuutta laajemmin verrattuna mallin rakentamiseen, koska hallittavuuden arviointia ei käsitellä ketterän kehityksen menetelmiä käsittelevässä kirjallisuudessa riittävästi tarkastelun toteuttamiseen.

Etsin mallin rakentamisessa käyttämääni kirjallisuutta pääosin muilta tieteenaloilta, koska ketterää kehitystä on käsitelty asiakirjahallinnan tutkimuksessa niin vähän. Lähinnä hyödynnän tietojenkäsittelytieteen ja hallintotieteiden kirjallisuutta, joissa ketterien kehitysmenetelmien hyödyntämistä järjestelmäsuunnittelussa on käsitelty eniten. Olen etsinyt kirjallisuuden neljän työssä hyödynnetyn ketterän kehityksen menetelmän – osittamisen, prototyyppien, pilottiprojektien ja iteroinnin – pohjalta. Pääosin työssä käytetty kirjallisuus koostuu tämän takia yleisten ketterää kehitystä kuvaavien artikkeleiden lisäksi yksittäisiä menetelmiä ja niiden käyttöä kuvaavista artikkeleista.

Valitsin jokaista menetelmää kohti kolme kirjoitusta, joiden arvioin kattavimmin käsittelevän käsittelemieni menetelmien perustekniikoita. Ensimmäisen analyysikierroksen jälkeen valitsin vielä tarpeen mukaan maksimissaan kaksi menetelmiä käsittelevää kirjoitusta lisää. Kirjallisuuden valinnassa käsittelin prototyyppejä ja iterointia yhtenä kategoriana, koska niitä käsittelevä kirjallisuus limittyy artikkeleissa vahvasti toisiinsa ja kaikki näitä menetelmiä hyödyntävät perusteokset käsittelevät molempia menetelmiä. Samoin pilottiprojektit ja jatkuva arviointi ovat yhdessä, koska käsittelen jatkuvaa arviointia vaihtoehtona pilottiprojektien toteuttamiselle ja käyttäjien tuomiselle vahvemmin suunnitteluun ja näitä käsittelevä kirjallisuus on tämän takia paljolti päällekkäistä. Peruskirjoitusten lisäksi hyödynnän kaikkien menetelmien käsittelyssä DIRKS-ohjeistusta ja AMS-opasta mahdollisuuksien mukaan.

Käytetyt kirjoitukset menetelmittäin:
Osittaminen:

- Collyer, Simon & Warren, Clive & Hemsley, Bronwyn & Stevens, Chris 2010. Aim, Fire, Aim—Project Planning Styles in Dynamic Environments. Project Management Journal, September 2010, 108–121.

- Jain, Rashmi & Chandrasekaran, Anithashree 2009. Rapid System Development (RSD) Methodologies: Proposing a Selection Framework. Engineering Management Journal, vol. 21 no. 4, 30–35.
- Karlström, Daniel & Runeson, Per 2005. Combining Agile Methods with Stage-Gate Project Management. IEEE Software, vol. 2 no. 3, 43–49.

Pilottiprojektit ja jatkuva arviointi:

- Bansler, Jørgen & Havn, Erling 2010. Pilot implementation of health information systems: Issues and challenges. International Journal of Medical Information, vol. 79 no. 9, 637–648.
- Glass, Robert 1997. Pilot Studies: What, Why, and How. Journal of Systems and Software, vol. 36, no. 1, 85–97.
- Johnston, Gary 2004. Applying IS development techniques to improve the quality of Records Management systems. <http://www.audata.co.uk/component/option,com_docman/task,doc_download/gid,2/> .
- Kautz, Karlheinz 2011. Investigating the design process: participatory design in agile software development. Information technology & People, vol. 24 no. 3, 217–235.
- Remenyi, Dan & Sherwood-Smith, Michael 1999. Maximise information systems value by continuous participative evaluation. Logistics Information Management, vol. 12 no. 1–2, 14–31.

Prototyypit ja iteroiminen:

- Alavi, Maryam 1984. An assessment of the prototyping approach to information systems development. Communications of the ACM, vol. 27 no. 6, 556–563.
- Center for Technology in Government 1998. A Survey of System Development Process Models. <http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf>.
- Evans, Joanne & Rouche, Nadav 2006. Utilizing Systems Development Methods in Archival Systems Research: Building a Metadata Schema Registry. Archival Science, vol. 4, no. 3–4, 315–334.

- Rzevski, George 1984. Prototypes versus pilot systems: strategies for evolutionary information system development. Julkaistu teoksessa R. Budde, K. Kuhlenkamp, L. Mathiassen, L. Züllighoven (toim.). Approaches to Prototyping. Springer, Heidelberg, 341–356.

Olen kirjallisuuden valinnassa pyrkinyt löytämään tasapainon sen välillä, että työ pysyy hallittavana ja toisaalta, että jokaisen menetelmän soveltamisessa ja vaikutusten arvioinnissa menetelmien perussoveltaminen tulisi käytyä riittävän perusteellisesti läpi. Vaikka olen valinnut menetelmää kohden tietyn määrän kirjallisuutta, useissa artikkeleissa käsitellään useampaa työssä sovelletuista menetelmistä ja siksi myös kirjallisuutta on sovellettu monin paikoin myös muihin menetelmiin kuin siihen, minkä alueelta kirjallisuus on valittu. Mallin rakentamiseen valitun kirjallisuuden lisäksi olen hyödyntänyt käsittelyssä myös muuta kirjallisuutta tarpeen mukaan, mutta kolme peruskirjoitusta ja lisäkirjoitukset muodostavat mallin rakentamisen ja asetettujen kriteerien täyttymisen analysoinnin pohjan.

Mallin vaikutusten arvioinnissa suunnittelun hallittavuuteen käytetyn kirjallisuuden olen valinnut joustavammin ja laajemmasta aineistosta, koska aihetta käsittelevää kirjallisuutta on vaikeampaa löytää ja yksittäiset teokset käsittelevät aihetta usein vain lyhyesti. Käytetty kirjallisuus koostuu ketterän kehityksen menetelmien vaikutuksia arvioivan kirjallisuuden ohella asiakirjahallinnan kirjallisuudesta ja erilaisista asiakirjahallinnan ja ketterän kehityksen casetutkimuksista, joissa käsitellään projektien toteuttamista ja niistä saatuja kokemuksia.

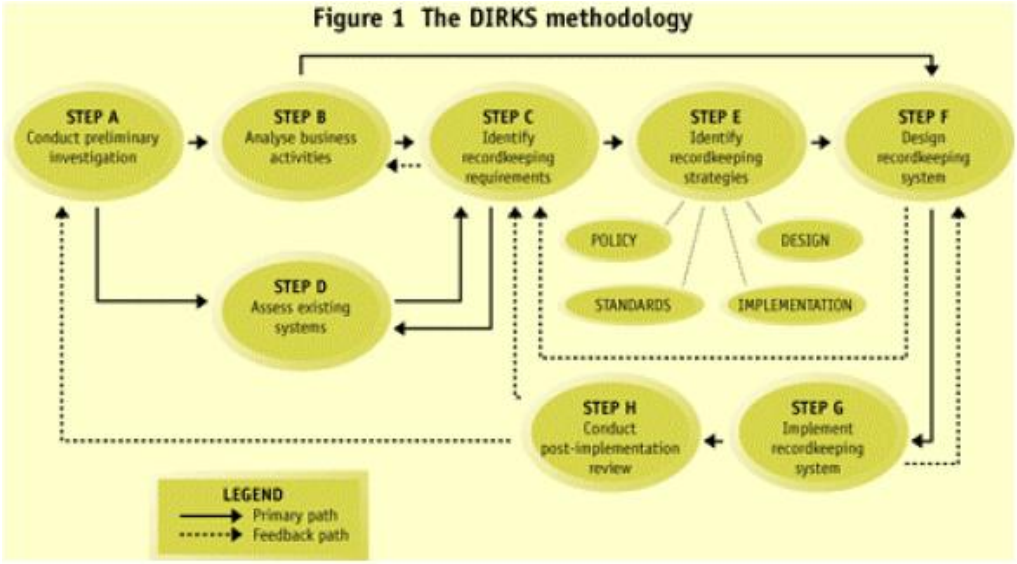
4 UUDEN MALLIN KEHITTELY

Tässä luvussa rakennan sähköisen asiakirjajärjestelmän tietosisällön suunnittelumallin. Luvun ensimmäisessä alaluvussa 4.1 esittelen tarkemmin DIRKS-ohjeistuksen ja AMS-oppaan sekä rakennan niiden pohjalta mallille rungon, jossa on kuvattu keskeiset tietosisällön tuottamisen vaiheet ja ketterien kehittämismenetelmien soveltamisen kannalta tarpeelliset yleiset suunnittelun toteutuksen vaiheet. Rungon rakentamisen jälkeen sovellan seuraavissa alaluvuissa yksi kerrallaan valitsemiani ketterän kehityksen menetelmiä mallin runkoon ja arvioin menetelmien soveltuvuutta tietosisällön suunnitteluun sen perusteella, miten ne vaikuttavat suunnittelun nopeuteen ja joustavuuteen. Kun olen tarkastellut yksi kerrallaan ketterien kehitysmenetelmien soveltamista malliin, kokoan näistä yksittäisistä vaiheista viimeisessä alaluvussa suunnittelun kokonaismallin, jossa sovitan eri menetelmät yhdeksi kokonaisuudeksi.

4.1 Mallin perusrakenne

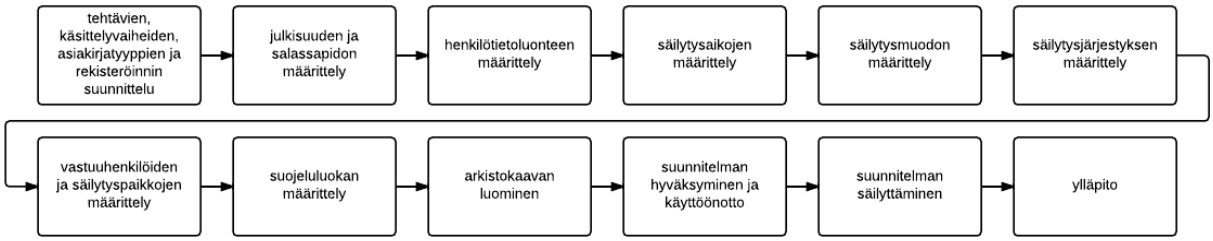
4.1.1 DIRKS, AMS-opas ja niiden yhteys

DIRKS-ohjeistuksessa kuvataan, kuinka organisaatio voi suunnitella asiakirjahallinnan strategian ja sen toteuttamiseen vaadittavat välineet sekä huomioida muut strategian toteuttamisen kannalta keskeiset tekijät, kuten käyttäjät. DIRKS:n asiakirjahallinnan suunnittelun kokonaisuudessa asiakirjahallinnan välineiden suunnittelu on vain yksi vaihe:



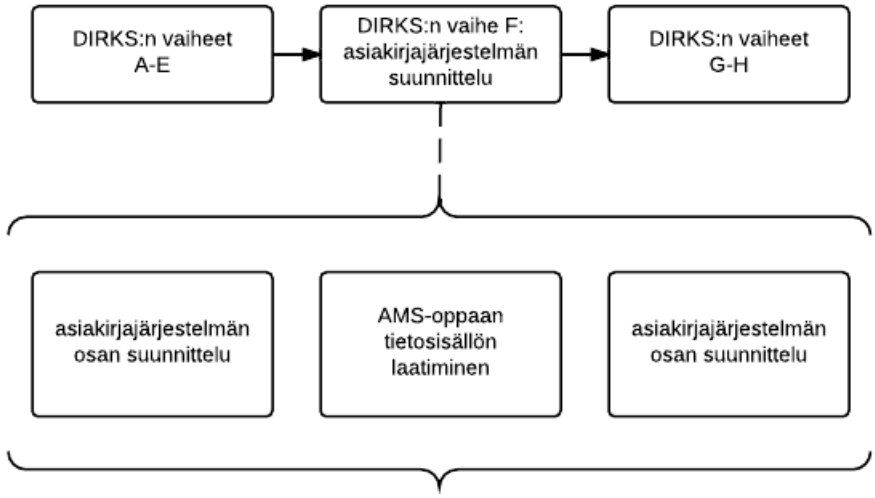
Kuvio 2. DIRKS:n vaiheet (National Archives of Australia 2001, 13, a user's guide)

DIRKS-mallissa varsinainen asiakirjahallinnan välineiden suunnittelu tapahtuu vasta vaiheessa F, jossa suunnitellaan asiakirjajärjestelmä, joka DIRKS:n tapauksessa tarkoittaa myös asiakirjojen hallinnan välineiden suunnittelua. Tietosisällön suunnittelu jakaantuu mallissa useampaan paikkaan, koska liiketoimintojen analysointi toteutetaan jo B-vaiheessa vaiheen C suorittamista varten. AMS-oppaassa tietosisällön tuottaminen on puolestaan kuvattu yhtenä toisiaan seuraavista vaiheista koostuvana kokonaisuutena:



Kuvio 3. AMS-oppaan etenemiskaavio

DIRKS-malliin sijoittuna tämä tietosisällön suunnittelu olisi luontevinta sijoittaa vaiheen F alle, jossa asiakirjahallinnan välineet suunnitellaan. AMS-oppaan vaiheet eivät sinällään olisi näin suoraviivaisesti sovellettavissa, koska DIRKS:ssä liiketoimintojen analysointi suoritetaan aiemmin ja myös AMS-oppaan loppuvaiheen käyttöönoton ja ylläpidon vaiheet tulevat DIRKS:ssä vasta seuraavissa vaiheissa. AMS-oppaan ja DIRKS:n yhteensopivuus on kuitenkin oman työnsä aihe ja kysymys ei ole olennainen tämän työn kannalta, koska mallia ei ole tarkoitus rakentaa kummankaan ohjeistuksen kanssa yhteensopivaksi, vaan hahmottaa vain niiden pohjalta tarpeelliset vaiheet. Karkeasti AMS-opas voisi kuitenkin DIRKS:iin piirrettynä sijoittua näin:



Kuvio 4. AMS-oppaan tietosisällön suunnittelun paikka DIRKS-mallissa

Tietosisällön laatiminen on yksi pieni osa DIRKS:n kokonaisuutta ja asiakirjajärjestelmän suunnittelussakin vain yksi järjestelmän suunnittelun osista. Lisäksi tietosisällön laatiminen on vain sähköisen asiakirjajärjestelmän suunnittelun osa, ei kokonainen F-vaiheen kokonaisuus itsessään.

Sähköisen asiakirjajärjestelmän tietosisällön suunnittelun kannalta konkreettiset suunnitteluvaiheet tapahtuvat siis lähinnä AMS-oppaan kuvaamissa vaiheissa. Myös DIRKS:ssä kartoitetaan vaiheessa B organisaation liiketoiminnot, joka vastaa AMS-oppaan tehtävien ja käsittelyvaiheiden kartoittamista. Molemmissa ohjeistuksissa myös laaditaan tehtäväluokitus tämän kartoituksen pohjalta. Muuten DIRKS ei ota kantaa sähköisen asiakirjajärjestelmän tai minkään asiakirjahallinnan välineen tietosisällön tuottamisen yksityiskohtiin.

4.1.2 Mallin runko

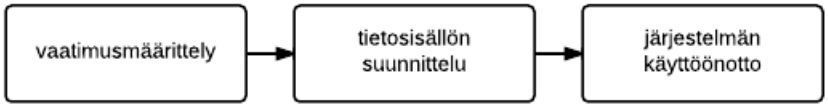
Tietosisällön tuottamisen vaiheet voi siis ottaa edellisen luvun käsittelyn perusteella pelkästään AMS-oppaasta. Konkreettisen suunnittelun kannalta AMS-oppaan kuvaus alkaa kuitenkin ”tyhjästä” siinä mielessä, että se ei ota mitenkään kantaa siihen, mitä ennen tietosisällön suunnittelua tehdään. Tämä johtuu siitä, että AMS-opas pyrkii vain kuvaamaan tietosisällön kartoittamisen vaiheet, ei ottamaan kantaa käytännön toteutukseen.

Tämän työn suunnittelumallissa on kuitenkin tarpeellista esittää jonkinlainen lähtöpiste suunnittelulle, jossa määritellään suunnittelun tavoitteet ja tehdään muut konkreettista suunnittelua edeltävät toimet. DIRKS:ssäkään ei oteta kantaa tarkasti siihen, miten eri vaiheiden aloitusvaihe toteutetaan, mutta DIRKS:n esikartoitusvaihe A on itsessään eräänlainen laajempi suunnittelun käynnistämisvaihe ja sen soveltaminen myös pienemmässä mittakaavassa tietosisällön tuottamisessa on hyvä lähtökohta. DIRKS:n vaiheessa A kerätään tarvittavaa taustatietoa projektin tarpeisiin, kartoitetaan organisaation vahvuuksia ja heikkouksia sekä luodaan johdolle suunnitelma projektin toteuttamisesta (National Archives of Australia 2001, 4, 2A). Käytännössä esikartoituksessa selvitetään asiakirjahallinnan nykytilanne ja arvioidaan järkeviä kehittämiskohteita. Esisuunnittelun pohjalta voidaan arvioida onko sähköisen asiakirjajärjestelmän käyttöönotto tai kehittäminen edes mielekästä organisaation tarpeisiin ja mitä tavoitteita kehittämiselle pitäisi asettaa.

Tietosisällön suunnittelun esisuunnittelun ei tarvitse olla tietosisällön suunnittelun kaltaisessa rajatummassa kokonaisuudessa näin kattavaa. Ketterien kehitysmenetelmien soveltamisen kannalta on järkevää pitää yksittäisiä iteraatioita koskevat etukäteissuunnittelut kevyinä. Esimerkiksi Alavi (19–20) on todennut, että vaatimusmäärittelyvaiheen on tarkoitus olla ketterässä kehittämisessä kevyt, että suunnittelu voidaan aloittaa nopeasti. Voimakkaasti iteratiivisissa prosesseissa tällainen kevyt vaatimusmäärittely on hänen mukaansa tärkeää. Collyerin et al. (2010, 109) mukaan iteratiivisessa kehittämisessä ajatuksena on, että suunnittelu aloitetaan vain hyvin yleisen tason kehyssuunnitelmasta ja suunnitelman yksityiskohtia täydennetään suunnitellun edetessä sitä mukaa, kun niistä opitaan enemmän. Suunnittelun yksityiskohtia voidaan hankkia esimerkiksi testaamisella, prototyypeillä, pilottiprojekteilla ja rinnakkaisilla kokeilla eri lähestymistavoista.

Silti myös tietosisällön ketterässä kehittämisessä on DIRKS:n esisuunnittelun tavoin arvioitava esimerkiksi suunnittelun etenemistä palautteen pohjalta, tarkasteltava kehittämiskohteita ja määriteltävä taustatietojen pohjalta tavoitteet suoritettavalle suunnittelukokonaisuudelle. Jonkinlainen suunnittelua edeltävä vaatimusmäärittely on käytännössä aina tarpeellinen, että suunnittelua voidaan ohjata jotain tavoitetta kohti. Tietosisällön suunnittelussa esitutkimustermi ei ehkä kuvaa kovin hyvin nopeaa vaatimusten määrittelyä ja muita suunnittelua ohjaavia toimenpiteitä. Tämän takia käytän termiä vaatimusmäärittely kuvaamaan yksittäisen suunnittelukokonaisuuden alussa suoritettavaa aloitusvaihetta.

Suunnittelun alkuvaiheen lisäksi asetan mallissa suunnittelun tarkastelulle myös konkreettisen päätepisteen, koska tämän avulla on selkeämpää hahmottaa, miten tietosisällön suunnittelu sijoittuu laajempaan sähköisen asiakirjajärjestelmän suunnitteluun, johon kuuluvat muun muassa sähköisen asiakirjajärjestelmän tekninen toteuttaminen. DIRKS:ssä suunnittelu päättyy varsinaisesti järjestelmän käyttöönottoon päivittäisessä työssä. Tämä on looginen päätepiste myös tietosisällön tarkastelulle. En sinällään käsittele käyttöönottoa, eikä se kuulu varsinaiseen tietosisällön suunnitteluun, mutta se toimii selkeänä loppurajauksena tarkastelulle. Lisäksi esimerkiksi pilottiprojektit edeltävät usein juuri käyttöönottoa ja ne on selkeää piirtää suhteessa siihen. Käytännössä ketteriä kehitysmenetelmiä sovelletaan yleensä niin, että suunnittelu ei pääty järjestelmän käyttöönottoon, vaan jatkuu myös sen jälkeen. Otan tämän huomioon menetelmien soveltamisessa tarpeen mukaan, mutta kuvioissa rajaan tarkastelun loppumaan selvyyden vuoksi käyttöönottoon. Tällä tavalla ajateltuna malli rajautuisi siis näin:



Kuvio 5. Mallin alku- ja päätepisteet DIRKS:n pohjalta

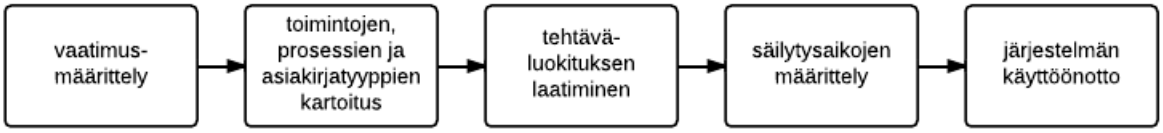
Varsinaiset tietosisällön kuvaamisen vaiheet valitsen AMS-oppaasta. Tämän työn kannalta AMS-oppaan kaikkien tietosisällön tuottamisen vaiheiden kuvaaminen sellaisenaan ei kuitenkaan ole mielekästä yksityiskohtaisuuden takia ja otan malliin vain keskeiset vaiheet. AMS-oppaassa tietosisällön suunnittelu alkaa tehtävien, käsittelyvaiheiden ja asiakirjatyyppien suunnittelusta. Tämä vaihe on keskeinen käytännössä aina asiakirjahallinnan suunnittelussa ja sisällytän sen siksi mallin runkoon. Asiakirjatyyppien kartoitus on AMS-oppaassa sidottu tehtävien ja prosessien kuvaamiseen ja siksi myös sen ottaminen mukaan selkeyttää tarkastelua. AMS-oppaan ja DIRKS:n etenemiskaavioissa ei ole piirretty näkyviin tehtäväluokituksen muodostamista tehtävien ja prosessien kartoituksen pohjalta, mutta sekin on vaihe, joka käytännössä aina suoritetaan suunnittelussa ja on loogista lisätä malliin tehtävien ja käsittelyvaiheiden suorittamisen ohella.

Tehtävien, prosessien ja asiakirjatyyppien kuvaamisen ohella kaikki muut AMS-oppaan kuvaamat tietosisällön tuottamisen vaiheet ovat erilaisten asiakirjojen hallinnan metatietojen tuottamista koskevia ja näitä metatietoja voidaan yhtälailla soveltaa myös sähköisessä asiakirjajärjestelmässä. Näitä metatietoja ovat esimerkiksi asiakirjojen säilytysajat, julkisuus- ja salassapitotiedot, säilytyspaikka, asiakirjojen henkilötietoluonne, säilytysmuoto ja suojeluluokka. Koska erilaisten asiakirjojen metatietojen suunnittelu ei eroa suunnitteluprosessina paljoa toisistaan, käsittelen metatietojen tuottamista mallissa tiivistettynä ottamalla malliin näkyviin vain säilytysaikojen määrittelyn. Lisäksi koska tarkoituksena on tarkastella yleisemmällä tasolla sitä, miten sähköisen asiakirjajärjestelmän tietosisällön suunnittelua voi tehostaa ketterillä kehitysmenetelmillä, metatietojen karsimisen lisäksi jätän mallista pois rekisteröinnin suunnittelun ja käsittelyprosessien ohjaamisen suunnittelun, joka on AMS-oppaassa esitetty sähköistä arkistonmuodostussuunnitelmaa käsittelevässä osiossa.

Pyrin rajauksilla pitämään mallin rungon yksinkertaisena ja yleisluontoisena, että pystyn selkeästi kuvaamaan ketterien kehitysmenetelmien soveltamista keskeisimpiin tietosisällön tuottamisen vaiheisiin. Tarkoitus on pitää ketterät kehitysmenetelmät tarkastelussa keskeisinä niin, että voin käsitellä niitä monipuolisesti sen sijaan, että kävisin yksityiskohtaisesti läpi kaikki mahdolliset tietosisällön tuottamisen vaihtoehdot ja kansalliset erityispiirteet. Mallis-

sani pyrin karsimaan tietosisällön tuottamisen sellaisiin vaiheisiin, joita sovelletaan useimmiten laadittaessa sähköistä asiakirjajärjestelmää.

Jättämällä tarkasteluun metatietojen tuottamisesta vain säilytysaikojen määrittelyn ja lisäämällä DIRKS:n pohjalta asetetut alku- ja päätepisteet, mallin lopullinen runko näyttää tältä:



Kuvio 6. Mallin suoraviivaistettu runko

Vaikka rungossa ei ole kuvattu kaikkien metatietojen suunnittelua, nekin voi ajatella sisältyvän malliin, koska työssä käytettyjä ketteriä kehitysmenetelmiä voi soveltaa myös niihin samoilla periaatteilla kuin säilytysaikojen määrittelyyn. Yksittäisten vaiheiden käsittelyn lisäksi kokonaisuuden tasolla tapahtuvat suunnitteluratkaisut ovat käsittelyssä keskeisiä ja otan näissä huomioon myös kuviossa näkymättömät vaiheet, jos niillä on vaikutusta menetelmien soveltamiseen.

4.2 Ketterän kehityksen menetelmien soveltaminen malliin

4.2.1 Projektin osittaminen

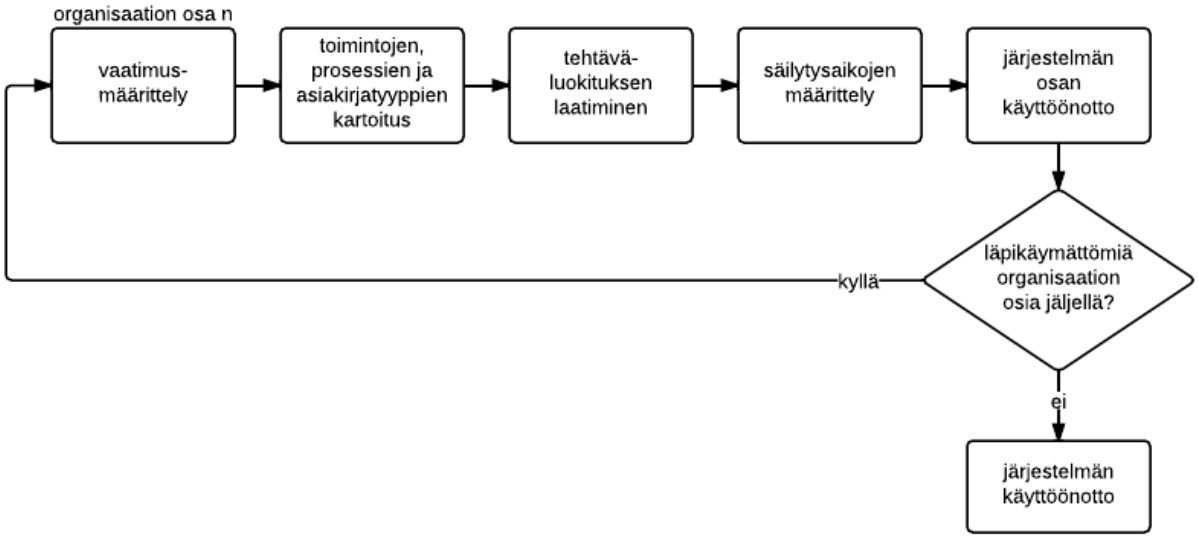
Tässä ja seuraavissa alaluvuissa sovellan erilaisia ketterän kehityksen menetelmiä edellisessä luvussa luotuun suunnittelumallin runkoon. Esittelen yksi kerrallaan kirjallisuudessa esitettyjä erilaisia tapoja soveltaa ketteriä kehitysmenetelmiä suunnittelussa ja arvioin sen jälkeen kirjallisuutta hyödyntäen soveltuvatko ne yleisesti sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun ja toisaalta miten ne vaikuttavat asettamiini soveltamisen kriteereihin: suunnittelun nopeuteen ja joustavuuteen.

Suunnittelun pilkkominen osiin on yksinkertaisimpia tapoja ketteröittää järjestelmien suunnittelua. Kutsun tätä prosessia työssä osittamiseksi. Aloitan ketterien kehitysmenetelmien soveltamisen osittamisesta, koska suunnittelun osittaminen pienempiin kokonaisuuksiin vaikuttaa eniten siihen, miten muita ketterän kehityksen menetelmiä voi soveltaa malliin. Projektin osittamisen tavoitteena on jakaa suunnittelu pienempiin osiin niin, että yksittäiset suunnittelun

osat ovat hallittavampia kuin laajemman kokonaisuuden. Myöhemmissä luvuissa käsitellään sitä, miten suunnittelua voidaan joustavoittaa ja nopeuttaa esimerkiksi tekemällä useita kierroksia, eli iteraatioita saman suunnittelukokonaisuuden toteuttamiseksi. Tämäkin on tavallaan tapa jakaa projekti hallittavampiin ja pienempiin kokonaisuuksiin, mutta tässä luvussa käsitellään sitä, miten suunnittelua voidaan pilkkoa yhden koko järjestelmän suunnittelukierroksen aikana pienempiin kokonaisuuksiin.

Käytännössä jokainen vähänkään suurempi suunnitteluprojekti on ositettu. Esimerkiksi vesiputousmallissa, jossa edetään vaiheesta vaiheeseen, työ on ositettu toisistaan erillisiin vaiheisiin, jotka seuraavat toisiaan, ja seuraavaan vaiheeseen siirrytään, kun edellinen vaihe on suoritettu. Tällainen vaiheisiin jako on luonnollinen ja sitä sovelletaan normaalisti myös ketterässä kehityksessä.

Osittamista voi kuitenkin suorittaa myös monilla muilla tavoilla. Yksi mahdollisuus on se, että koko suunnitteluprosessi jaetaan pienempiin osiin organisaation osittain esimerkiksi seuraavasti:



Kuvio 7. Suunnittelun osittaminen organisaation osiin

Kuviossa n-kirjain kuvaa sitä organisaation osaa, jonka tietosisältöä kyseisellä kierroksella suunnitellaan. Suunnittelun alussa ensimmäisellä kierroksella käydään kaikki suunnittelun vaiheet läpi organisaation ensimmäiselle osalle (n = 1). Toisella kierroksella käydään läpi toisen (n = 2) organisaation osan kohdalta kaikki suunnitteluvaiheet läpi. Näin edetään niin kauan, että kaikki organisaation osat on käsitelty.

Tällaista lähestymistapaa voi muokata niin, että järjestelmä otetaan käyttöön vasta sitten, kun kaikkien organisaation osien järjestelmät ovat valmiita. Toinen mahdollisuus on tehdä ensimmäinen organisaation osa valmiiksi, ottaa järjestelmä käyttöön tässä osassa ja toteuttaa vasta sen jälkeen loput organisaation osat rinnakkain suoritettuna saatujen kokemusten pohjalta. Tällaista lähestymistapaa käytetään esimerkiksi hajaantuneissa organisaatioissa, joissa toiminnot voivat sijaita vaikkapa eri maissa (Collyer et al. 2010, 115). Tällä tavalla suoritettuna ensimmäisen organisaation osan suunnitteluprojekti paitsi edistäisi projektia myös muodostaisi eräänlaisen pilottiprojektin myöhempien osien suorittamiselle. Pilottiprojekteja käsitellään tarkemmin myöhemmin, mutta on syytä tunnistaa jo tässä vaiheessa, että projektin osittaminen muodostaa periaatteessa pilottiprojekteja, vaikka niitä ei erikseen suunniteltaisikaan.

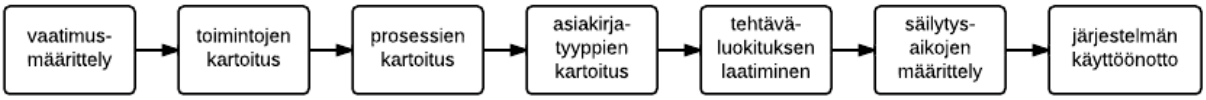
Työn jakaminen organisaatioyksiköittäin voi olla hyödyllistä myös siksi, että se mahdollistaa halutunlaisen käyttäjäryhmän valitsemisen suunnitteluprojektin alkuvaiheeseen. Voidaan esimerkiksi etsiä keskeinen organisaation osa, jota asiakirjahallinnan kehittäminen erityisesti koskee tai sellainen organisaation osa, jonka oletetaan suhtautuvan myönteisesti asiakirjahallinnan kehittämiseen ja jolta oletetaan saatavan helpommin palautetta projektin edistyessä. Esimerkiksi prototyyppien testaaminen ja pilottiprojektien toteuttaminen vaatii käyttäjien saamista mukaan aktiivisesti suunnittelutyöhön. Tämä on helpompaa, jos suhtautuminen suunnitteluun koskettaa käyttäjiä merkittävästi tai he ovat muuten myönteisiä uudistuksille.

Myönteisen suhtautumisen merkitys ensimmäisissä organisaation osissa korostuu, koska prototyyppien ja pilottiprojektien hyödyntäminen on keskeistä projektin alkuvaiheessa, kun tarkkaa käsitystä suunnittelun vaatimuksista ei vielä ole. Vähämerkityksisempien organisaation osien kohdalla voidaan sitten alussa saatujen kokemusten pohjalta keventää prototyyppien ja pilottiprojektien käyttöä. Ensimmäisen organisaation osan suunnittelukokonaisuuden jälkeen loput organisaation osat voitaisiin suorittaa suurempina kokonaisuuksina ja suoraviivaisemmalla suunnittelumallilla kuin tässä työssä on esitetty. Esimerkiksi Di Biagio ja Ibiricu (2008, 172) ovat tutkineet sähköisen asiakirjajärjestelmän käyttöönottoa ja heidän esimerkissään käyttöönotto toteutettiin rinnakkain niin, että kahdesta kolmeen eri organisaation osaa toteutettiin kerralla.

Toinen mahdollisuus osittaa projektia on erilaisten yksittäisten vaiheiden tai vaiheiden muodostamien osakokonaisuuksien suorittaminen valmiiksi yksi kerrallaan omina projekteinaan.

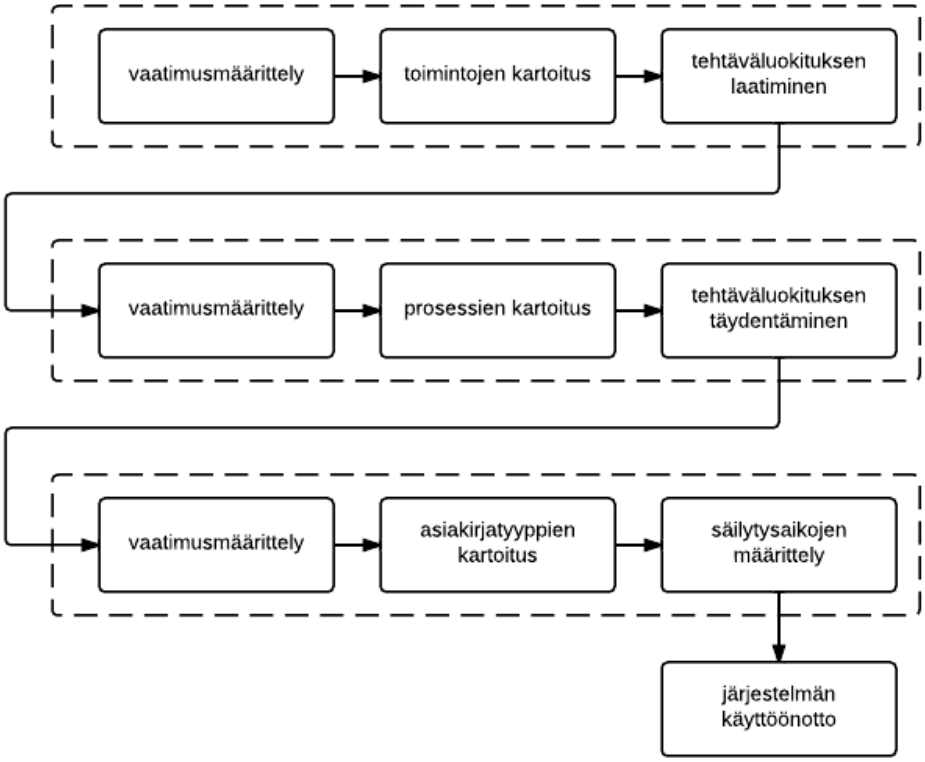
Tämänkaltaiset ositusmenetelmät ovat tyypillisiä evolutionaariselle kehittämiselle. Evolutionaarisissa kehittämismenetelmissä työ vaiheistetaan erilaisiin kokonaisuuksiin, joista jokainen toteuttaa jonkin järjestelmän osa-alueen. Tavoitteena tällaisessa lähestymistavassa on paitsi projektin pilkkominen hallittavampiin kokonaisuuksiin, myös käyttäjäpalautteen maksimointi. Suoritettavat osatkin valitaan monissa evolutionaarisen kehittämisen malleissa siinä järjestyksessä, että asiakkaalle kaikkein tärkeimmät ja näkyvimmät osat järjestelmästä suunnitellaan ensin. Vaikka ajatusta ei vietäisikään näin pitkälle, asiakaspalautteen kerääminen on tällaisessa lähestymistavassa keskeistä. (Jain & Chandrasekaran 32–33)

Sähköisen asiakirjajärjestelmän suunnittelussa ongelmana se, että mallista on vaikeaa löytää DIRKS:iin ja AMS-oppaaseen perustuvassa perusmuodossaan selkeitä kokonaisuuksia. Jos erikseen suoritettavia osia jaottelee tähän asti luodun mallin pohjalta, niistä tulee suuria, koska perinteisissä sähköisen asiakirjajärjestelmän pohjalle tietosisällönlaatimismalleissa suoritetaan perusteellinen tehtäväkartoitus ennen seuraavaan vaiheeseen siirtymistä. AMS-oppaan mallissa tehdään tämän lisäksi vielä asiakirjatyyppien kartoitus. Yksi mahdollisuus olisi pilkkoa tehtävien, prosessien ja asiakirjatyyppien kartoitus osiin seuraavasti:



Kuvio 8. Tehtävien, prosessien ja asiakirjatyyppien avaaminen omiksi vaiheiksi

Tämän jälkeen voidaan paremmin muodostaa yksittäisiä osakokonaisuuksia suoritettavaksi. Koska tavoitteena on maksimoida mallin välitavoitteiden nopeus ja joustavuus, kokonaisuudet on pidettävä pieninä. DIRKS-oppaassa sanotaan, että tehtäväluokituksen laatimisessa voidaan edetä kahdesta suunnasta, ylhäältä alas ja alhaalta ylös, mikä tarkoittaa, että tehtäväluokitusta rakennetaan sekä kartoittamalla yleisellä tasolla organisaation toimintoja että kuvaamalla alhaalta päin prosesseja, mikä paljastaa tehtävärakenteita (National Archives of Australia 2001, 6, osa 2B). Asiakirjatyyppien kartoittaminen ei ole tehtäväluokituksen laatimisen kannalta välttämätöntä, joten sen voi pudottaa ensimmäisestä kokonaisuudesta pois. Sen sijaan säilytysaikojen määrittely vaatii asiakirjatyyppien tunnistamista, jolloin näiden kahden yksittäisen vaiheen yhdistäminen voisi olla järkevää. Tällä tavalla muodostettuna malli voisi olla seuraavanlainen:



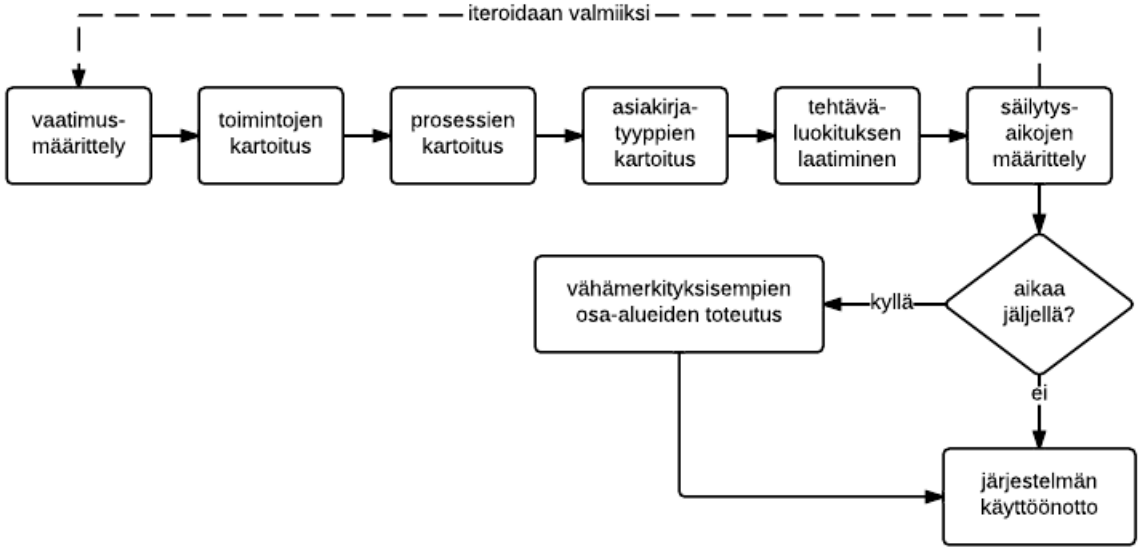
Kuvio 9. Suunnittelun jakaminen evolutionaarisen mallin mukaan osakokonaisuuksiin

Näin suunnittelu jakaantuu kolmeen erilliseen osakokonaisuuteen, joista jokaisessa suunnitellaan jokin yksittäinen tietosisällön osa. Kaksi ensimmäistä osakokonaisuutta johtavat tosin tällä tavalla jaettuna molemmat tehtäväluokituksen laatimiseen. Niissä kuitenkin toteutetaan selkeästi erilainen työvaihe. Lisäksi koska sekä toimintojen kartoitus että prosessien kartoitus ovat laajoja työvaiheita, ne on järkevää pitää erillään. Liian suurten osakokonaisuuksien muodostaminen ei ole asettamillani kriteereillä järkevää myöskään siksi, että se hidastaisi välitavoitteiden saavuttamista. Tämäkään jako ei ole välttämättä järkevä suunnittelun edistyessä esimerkiksi organisaation osasta toiseen. Kokemuksen ja tiedon lisääntyessä ja aiemmista organisaation osista saatujen tietojen pohjalta eri osakokonaisuuksia voi olla järkevää suorittaa tiiviimmin niin, että esimerkiksi toimintojen kartoitusta, prosessien kartoitusta ja asiakirjatyyppien määritystä tehdään rinnakkain. Suunnittelun alussa kokonaisuudet ovat kuitenkin jo yksinään niin laajoja, että eteneminen osakokonaisuuksittain on usein järkevää, vaikka se hidastaa etenemistä ja vaatii useampia työvaiheita.

Projekti voitaisiin osittaa myös niin, että kaikkein tärkeimmät osakokonaisuudet toteutettaisiin ensin. Tällaista lähestymistapaa hyödynnetään usein ohjelmistoteollisuudessa, jossa asiakkaan kannalta tärkeimmät ominaisuudet toteutetaan ohjelmistoon ensimmäisenä. Näin varmiste-

taan, että tärkeimmät osat tulevat valmiiksi, ja jos aikataulussa pysymiseksi joudutaan pudottamaan ominaisuuksia, ne ovat vähäpätöisempiä. Lisäksi tällä tavalla saadaan kerättyä aikaisemmin palautetta keskeisimmistä osista. (Karlström & Runeson 2005, 46) Sähköisen asiakirjajärjestelmän suunnittelussa tiettyjen yksittäisten suunnitteluvaiheiden suunnittelu ensin ei kuitenkaan ole mielekästä, koska eri vaiheet ovat riippuvaisia toisistaan. Esimerkiksi tehtäväluokituksen laatimisen säilytysaikojen määrittelyn välillä on vaikeaa sanoa, kumpi olisi tärkeämpää. Kaikki typistetyssä mallissa esitetyt osa-alueet ovat niin keskeisiä sähköisen asiakirjajärjestelmän toiminnan kannalta, että ne on toteutettava valmiiksi ennen käyttöönottoa.

Sen sijaan mallista pois jätettyjä erilaisten elinkaaren hallinnan metatietojen suunnittelua voisi ajatella jätettäväksi suunnittelussa myöhemmäksi. Esimerkiksi suojeluluokan määrittely ei yleensä ole yhtä keskeistä järjestelmän käyttöönoton kannalta kuin säilytysaikojen määrittely. Tällöin priorisoitaisiin tehtäväluokitus ensisijaiseksi ominaisuudeksi ja määriteltäisiin suojeluluokat vasta, kun säilytysajat on määritelty. Koska mallin runkoon valittujen suunnitteluvaiheiden kohdalla priorisointi ei vaikuta mielekkäältä, en lisää tätä erillistä suunnittelua omaan kokonaismalliini, mutta priorisointi voisi tapahtua esimerkiksi näin:

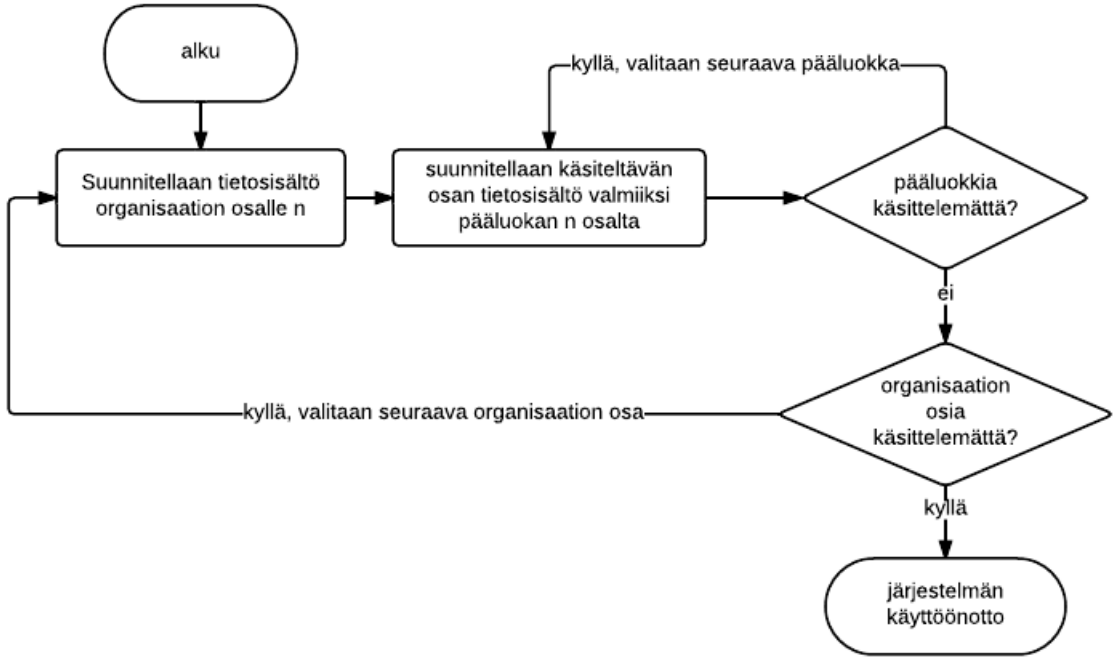


Kuvio 10. Tärkeimpien suunnitteluvaiheiden priorisointi

Vaikka priorisointi ei vaikuta keskeisiin vaiheisiin, myös keskeisten vaiheiden osalta olisi mahdollista soveltaa pilkkomista yksittäisten vaiheiden sisällä. Arkistolaitoksen eAMS-käyttöönottosuunnitelmassa on mainittu mahdollisuus suunnitella sähköistä asiakirjajärjestelmää tehtäväluokka kerrallaan (Arkistolaitos 2010). Tällä tavalla pilkottuna projektin yksittäisestä kokonaisuudesta voitaisiin tehdä pienempi ja hallittavampi. Esimerkiksi prosessien kartoitus sekä asiakirjatyyppien ja säilytysaikojen kartoitus voitaisiin tehdä pääluokka tai jopa

alemman tason luokka kerrallaan. Jakaminen pääluokkaa pienempiin kokonaisuuksiin olisi kuitenkin jo todella pieni kokonaisuus. Siihen tuskin olisi enää mielekästä soveltaa esimerkiksi pilottiprojekteja tai prototyypittämistä. Korkeintaan yksittäisten tehtäväluokkien toteuttamista voisi hyödyntää aivan projektin alussa, jos halutaan saada nopeasti tietoa projektin toteuttamisesta esimerkiksi vaatimusmäärittelyjä varten tai kokemuksen keräämiseksi jostain suunnittelun vaiheesta. Koska pilottiprojektien ja prototyyppien toteuttaminen näin pienelle kokonaisuudelle ei olisi mielekästä, palautteen saaminen käyttäjiltä olisi kuitenkin vähäistä ja siksi näin tarkka pilkkominen ei ole yleensä mielekästä.

Sen sijaan pääluokkien tasolla pilkkominen voisi olla vaihtoehto, jos suunnittelukokonaisuudet halutaan pitää pieninä ja hallittavina. Pääluokat muodostavat yleensä niin suuren kokonaisuuden, että pilottiprojektien ja prototyyppien käyttäminen niiden pohjalta tapahtuvassa suunnittelussa olisi mielekkäämpää. Tosin koko projektin toteuttaminen pääluokka kerrallaan olisi sekavaa ja pirstoisi useissa tapauksissa työtä liikaa, mutta erityisesti projektin alkuvaiheessa muutaman pääluokan toteuttaminen ennen laajemman kokonaisuuden suunnittelua kokemuksen ja palautteen saamiseksi voisi olla perusteltua. Tällainen jakaminen myös nopeuttaisi välitavoitteiden saavuttamista ja käyttäjäpalautetta voitaisiin kerätä pienemmistä kokonaisuuksista nopeammin. Yhdistettynä organisaation osien mukaan jakamisella suunnittelukokonaisuuksien koko pienenisi merkittävästi:



Kuvio 11. Suunnittelun osittaminen pääluokittain ja organisaation osittain

Näin voimakas osittaminen ei kuitenkaan ole koko tietosisällön suunnittelun mittakaavassa mielekästä, koska tehtäväluokituksen pääluokkien määrä liikkuu usein kymmenen molemmin puolin ja organisaation osia voi olla kymmeniä, jolloin erillisten osaprojektien määrä kasvaa helposti lähemmäs sataa. Tämän takia näin yksityiskohtaista jakamista on järkevää soveltaa korkeintaan aivan projektin alkuvaiheessa, kun halutaan kokemusta projektin toteuttamisesta ja palautetta rajatummasta kokonaisuudesta nopeasti. Toinen mahdollisuus on pilkkoa yksittäisten päätehtäväluokkien ja pienten organisaation osien sijaan kokonaisuuksia laajemmiksi. Esimerkiksi hallintoon liittyvät pääluokat voisi käsitellä yhtenä kokonaisuutena ja organisaation osista yhdistää vaikkapa erilaiset tukitoiminnot. Tällaiset valinnat ovat tasapainoilua sopivan pienien ja hallittavien kokonaisuuksien muodostamisen ja kohtuuttoman pirstaloitumisen välillä.

Pääluokittain osittamisessa on ongelmana myös se, että se luokittain osittaminen on mielekästä vain, jos sen pohjana on jo olemassa suhteellisen vakaa luokitus. Mallissani luokituksen teko on kuitenkin osa iteratiivista suunnittelua ja sitä parannetaan vaiheittain muun suunnittelun ohella. Tehtäväluokitus muotoutuu mallissa hiljalleen tietosisällön suunnittelun iteraatioiden suorittamisen myötä. Tämän takia pääluokittain osittamisen pohjaksi ei suunnittelun ensimmäisillä iteraatiokierroksilla olisi usein riittävän valmista luokitusta. Pääluokittain osittaminen toimisi paremmin, jos luokitus suunniteltaisiin pääosin valmiiksi ennen muun tietosisällön suunnittelua. Pääluokittain osittamisen olisi toki mahdollista pohjautua ensimmäisellä iteraatiokierroksella luotuun alustavaan luokitukseen. Tässä on kuitenkin riskinä se, että osittamisen pohjana oleva luokitus muuttuisi myöhemmin merkittävästikin, mikä vaikeuttaisi osittamisen suunnittelua ja hallintaa. Toinen vaihtoehto olisi vähentää mallin iteratiivisuutta ja suunnitella luokitus ensin mahdollisimman valmiiksi ja vasta sen jälkeen suunnitella muu tietosisältö. Tämä vähentäisi kuitenkin mallini kriteerien vastaisesti suunnittelun joustavuutta ja hidastaisi välitavoitteiden saavuttamista.

Näiden haasteiden takia pääluokittain osittaminen toimisi mallini rakenteella ja kriteereillä paremmin niin, että sitä hyödynnetään vain suunnittelun alkuvaiheessa, kun halutaan jakaa suunnittelua pieniin osiin esimerkiksi testien tekemiseksi tai kokemuksen hankkimiseksi. Vaikka kokonaisuutena tehtäväluokitus voi muuttua merkittävästi iteratiivisessa kehittämisessä vaiheiden välillä, jo ensimmäisen tehtäväluokituksen suunnittelun iteraation jälkeen on todennäköisesti mahdollista tunnistaa muutamia niin keskeisiä tehtäviä, että voidaan luottaa siihen, että ne säilyvät luokituksessa loppuun asti ja osittaa suunnittelua näiden pohjalta. Koko

suunnittelun osittaminen pääluokittain on kuitenkin ongelmallista suunnittelun pirstoutumisen ja iteratiivisen suunnittelun aiheuttaman tehtäväluokituksen muuttumisen vuoksi.

Vaikka se jää tästä mallista pois korkeamman käsittelytason takia, on myös huomattava, että ketteriin kehitysmenetelmien soveltamisen yhteydessä on yleistä, että suunnittelua pilkotaan osiin paitsi laajempien kokonaisuuksien osalta, myös yksittäisten suunnitteluvaiheiden sisällä konkreettisten työtehtävien suorittamisessa. Tavoitteena on jakaa työtä niin, että voidaan muodostaa helposti hallittavia yksittäisiä työtehtäviä, joihin voidaan keskittyä tehokkaasti. Tämä lisää suunnittelijoiden tunnetta siitä, että he hallitsevat suorittamiaan kokonaisuuksia ja vähentää sekaannusta eri tehtävien suorittamisessa. (Karlström & Runeson 2005) Myös Niemi-Grundström (2012) on korostanut tätä työn pilkkomisen tärkeyttä ketterien kehitysmenetelmien soveltamisen yhteydessä. Hänen mukaansa, kun eri suunnittelun vaiheet pilkotaan riittävän pieniin osiin, ne voidaan helpommin priorisoida ja aikatauluttaa ja pienet kokonaisuudet pysyvät helpommin hallittavina.

4.2.2 Pilottiprojektit

Pilottiprojekteilla tarkoitetaan järjestelmän testausprojekteja, joissa suunniteltua järjestelmää testataan työympäristössä niin, että järjestelmän loppukäyttäjät pääsevät käytännössä kokeilemaan järjestelmää ja antamaan palautetta siitä. Tarkoituksena pilottiprojekteissa on tunnistaa suunnitteluvirheitä ja muita potentiaalisia ongelmia ennen kuin järjestelmä otetaan käyttöön koko organisaatiossa. Pilottiprojektit ovat prototyyppien ohella tehokas tapa kerätä käyttäjiltä palautetta, koska niiden avulla käyttäjät voivat konkreettisesti nähdä, miten järjestelmä toimii. Usein järjestelmien suunnittelussa ja erityisesti lineaarisiin menetelmiin perustuvassa suunnittelussa palautetta kerätään käyttäjiltä esittämällä heille erilaisia abstrakteja suunnitelmia ja kaavioita, mutta tämä ei yleensä toimi, koska käyttäjät eivät pysty hahmottamaan järjestelmän toimintaa ilman käytännön kokeilua. Pilottijärjestelmien päätavoitteet liittyvät nimenomaan käyttäjäpalautteen keräämiseen. Piloteilla voidaan palautteen pohjalta arvioida järjestelmän hyödyllisyyttä ja parantaa suunnittelun toteuttamista. (Bansler & Havn 2010, 638–639)

Pilottiprojektit eivät ole vain yksi tietynlainen ja tietyssä vaiheessa suunnittelua suoritettava toimenpide. Vaikka pilottiprojektit usein toteutetaan projektin loppupuolella ennen järjestel-

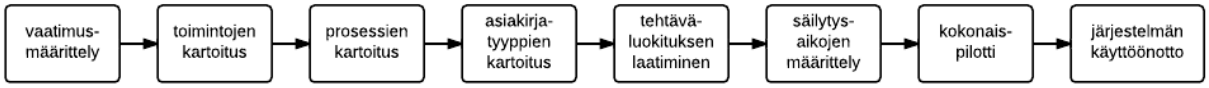
män käyttöönottoa ja niissä testataan koko järjestelmää, tämä ei ole ainoa lähestymistapa. Glass (1997, 85–86) on esittänyt, että pilottiprojekteja voidaan toteuttaa erilaajuisina sen mukaan, miten tärkeää toimintoa organisaation kannalta pilottiprojektilla testataan. Glass on jakanut pilottiprojektit kolmeen luokkaan niin, että kaikkein keskeisimpiä toimintoja testaavat projektit toteutetaan tiukalla (rigorous) toteutuksella, kaikkein vähämerkityksisimmät ja pienimmät kokonaisuudet käydään läpi epävirallisemmalla (informal) toteutustavalla ja välissä on vielä yksi luokka tärkeydeltään keskinkertaisille projekteille.

YLE:n arkisto-, medianhallinta- ja informaatiopalveluiden johtaja Katri Vänttinen (2012) on puolestaan esittänyt, että laajojen kokonaisten projektien testaamiseen tarkoitettujen pilottiprojektien ohella voidaan toteuttaa niin sanottuja minipilotteja, joissa testataan jotain pienempää kokonaisuutta epämuodollisemmin ja käytännönläheisesti muotoilemalla koeasetelman, jossa testataan jotain uutta ideaa ja sen toimivuutta käytännössä. Vänttisen minipilottiprojektin ajatus on hieman samankaltainen kuin Glassin (1997, 85–86) epävirallisen pienempien kokonaisuuksien testaamiseen tarkoitettu metodologia.

Usein pilottiprojektien ajatellaan kuitenkin olevan enemmän suunnittelun myöhäisempiin vaiheisiin kuuluva keskeinen suunnittelun väline, jolla pyritään varmistamaan, että kehitetty konsepti järjestelmän kehittämisestä todella toimii myös käytännössä. Suunnittelun ensimmäisten iteraatioiden kohdalla tiedonhankinta aiemmasta suunnittelusta siirtyy enemmän epämuodollisesti ja joustavasti, mutta suunnittelun lopussa vaaditaan pilottiprojektin kaltaista muodollisempaa ja järjestelmällisempää tapaa testata suunnittelua käytännössä. (Collyer et al. 2010, 115)

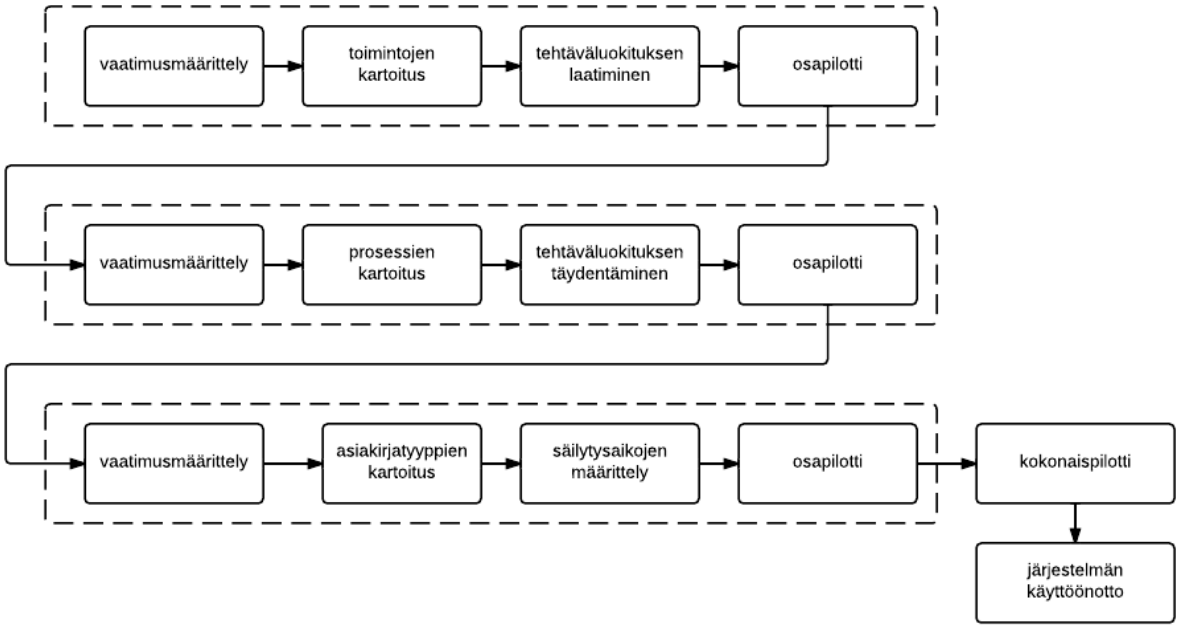
Glassin mukaan (1997) pilottiprojektien toteuttamisessa on paljon ongelmia ja hän on artikkelissaan esittänyt johdonmukaisemman mallin pilottiprojektien toteuttamiseen. Tämän työn laajuudessa ei ole mahdollista syvällisemmin esitellä erilaisia pilottiprojektien käytännön toteutuksen ongelmia, mutta sovellan kuitenkin Glassin ja Vänttisen ajattelun pohjalta malliin paitsi laajempia pilottiprojekteja, myös pienempien ja rajallisten minipilottiprojektien soveltamisen mahdollisuuksia eri vaiheiden sisällä.

Ilmeisin paikka lisätä suunnittelumalliin pilottiprojekteja on koko suunnittelukokonaisuuden testaaminen pilottiprojektilla. Tämä on myös pilottiprojektien yleisin käyttötapa:



Kuvio 12. Kokonaispilotin toteuttaminen suunnittelun lopussa ennen käyttöönottoa

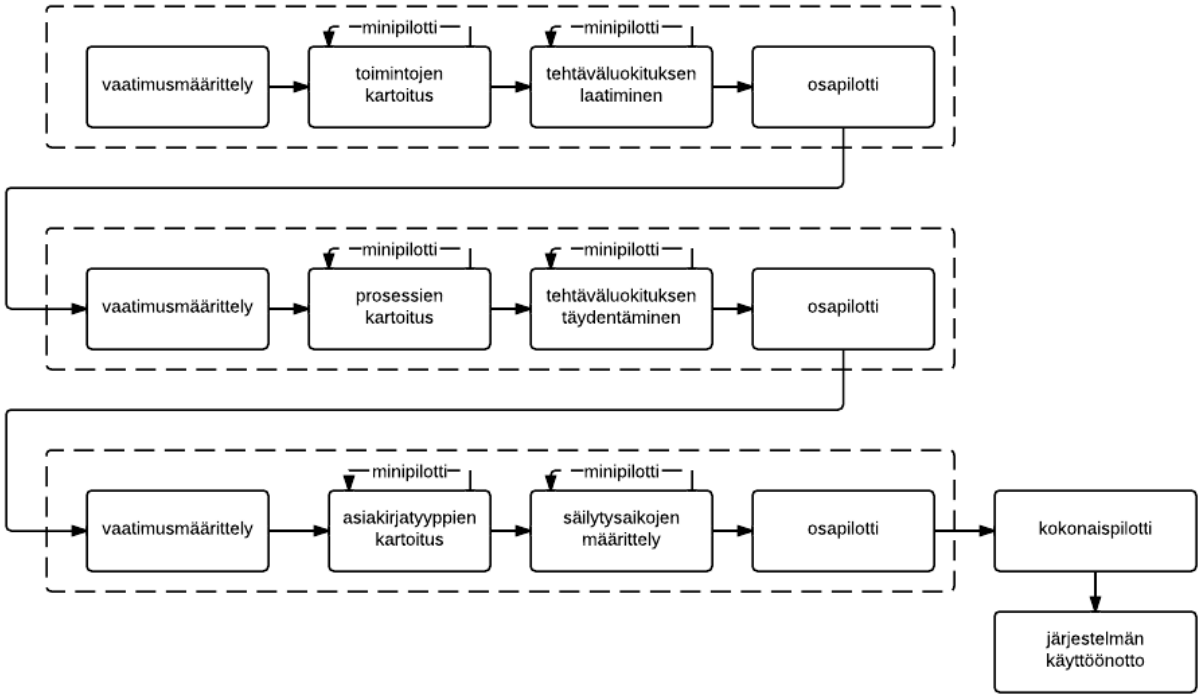
Tämä yksinään on vielä joustamaton malli ja tarjoaa vain vähän mahdollisuuksia kokeilla erilaisia lähestymistapoja ja kerätä niistä palautetta käyttäjiltä. Edellisessä luvussa esiteltiin yksi vaihtoehto siitä, miten malli voidaan osittaa evolutionaarisen kehittämismallin mukaisesti osakokonaisuuksiin. Nämä osakokonaisuudet muodostavat loogisen paikan lisätä pilottiprojekteja jokaisen osakokonaisuuden loppuun, jolloin voitaisiin testata pienempiä kokonaisuuksia kokonaispilotin ohella:



Kuvio 13. Osapilottien toteuttaminen osakokonaisuuksien lopussa

Näin myös osakokonaisuuksia voidaan testata ja kehittää osapiloteista saatujen tulosten ja palautteen pohjalta. Osakokonaisuuksien testaaminen piloteilla mahdollistaa monipuolisemman palautteen keräämisen, koska kerralla voidaan keskittyä rajatumpaan kokonaisuuteen ja toteuttaa pilotti kevyemmin kuin laajemmassa koko järjestelmää koskevassa pilotoinnissa. Tällainen evolutionaarisen mallin mukaisten osakokonaisuuksien testaaminen piloteilla on vaativuudeltaan luontevaa rinnastaa Glassin (1997, 86–88) esittämässä jaottelussa niin sanottujen keskivaativien tai jopa matalan tärkeyden pilottien joukkoon. Tällöin ne voidaan toteuttaa Glassin mallia seuraten kevyemmällä kaavalla kuin laajempaa ja tärkeämpää kokonaisuutta koskevat pilottiprojektit, mikä voi tarkoittaa esimerkiksi pilotin suunnittelun keventämistä ja tilastollisten menetelmien pudottamista pois tulosten tarkastelusta.

Pilottiprojekteja voidaan soveltaa järkevästi vielä yhtä tasoa alempana. Paitsi yksittäisiin suunnittelukokonaisuuksiin, pilotteja voidaan toteuttaa myös yksittäisissä suunnitteluvaiheissa. Tällainen lähestymistapa vertautuu Vänttisen (2012) minipilotteihin tai Glassin (1997, 86) epämuodollisten tai pienimpien pilottiprojektien toteuttamiseen. Jos minipilotit lisättäisiin suunnittelumallin yksittäisiin vaiheisiin, malli näyttäisi seuraavalta:



Kuvio 14. Minipilottien käyttäminen suunnittelun yksittäisissä vaiheissa

Ongelmana osapilottien ja erityisesti minipilottien lisäämisessä tällä tavalla on se, että vaikka joustavuus ja mahdollisuudet kerätä käyttäjiltä palautetta lisääntyvät, välitavoitteiden saavuttamisen nopeus kärsii merkittävästi. Jokainen toteutettu pilottiprojekti hidastaa osakokonaisuuksien ja koko järjestelmän iteraatioiden valmistumista. Pilottiprojektit ovat kuitenkin hyvä keino lisätä joustavuutta suunnitteluun ja niiden harkittu käyttö voi jopa nopeuttaa suunnittelua ja parantaa lopputulosta, kun päästään testaamaan joustavammin erilaisia suunnitteluvaihtoehtoja ja vältetään virheitä.

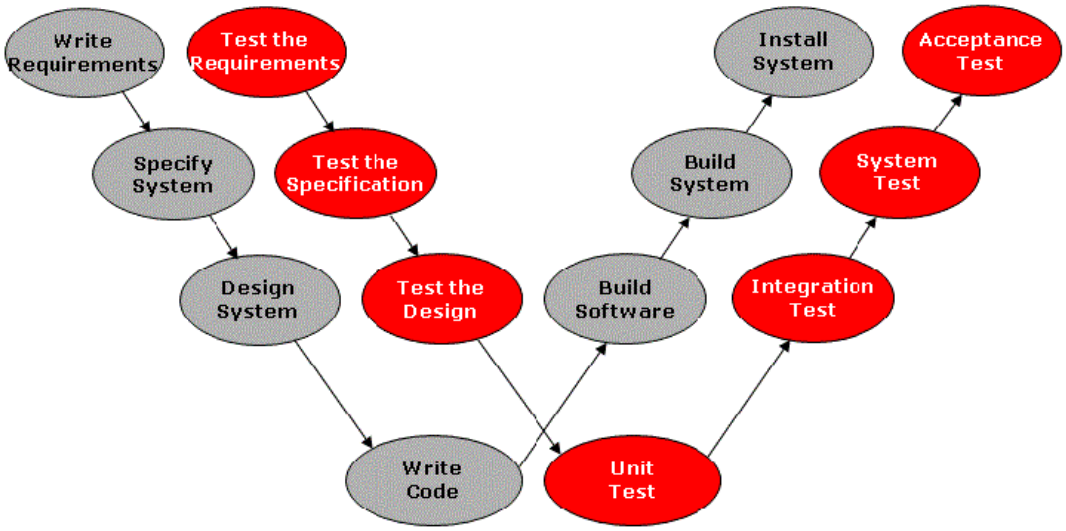
Pilottien toteutusta ei voi soveltaa mekaanisesti tietyn ennalta määritellyn kaavan mukaan. Mini- ja osapilottien toteuttamisen tarve täytyy punnita joka kerta erikseen. Esimerkiksi niissä vaiheissa, joiden toteuttamisesta projektiryhmällä on vähiten kokemusta tai jotka muuten ovat haastavia, voi olla järkevää toteuttaa pilottiprojekti. Myös lupaavien uusien lähestymistapojen tai yksittäisten ideoiden testaaminen piloteilla on hyvä esimerkki pilottien hyödyntämisestä. Toisaalta jokainen pilotti hidastaa laajempien kokonaisuuksien valmistumista ja riskinä on

jumittua liian pitkäksi aikaa yksittäisen vaiheen suunnitteluun. Tämä voi vähentää paitsi suunnittelun nopeutta myös sen joustavuutta, kun iteroinnille ei jää niin paljoa aikaa. Myös resurssit ovat rajallisia. Liiallinen ja perusteeton pilotointi on helposti pois muusta suunnittelusta.

Pilottien pitämiseksi hallittavina osakokonaisuuksien ja yksittäisten suunnitteluvaiheiden pilotointi on oltava kevyempää kuin kokonaisuuden testaamiseen käytetty pilotointi. Sanan pilotointi käyttäminen kaikista eri tason käytännön testeistä on tässä yhteydessä ehkä hämäävää, koska koko järjestelmän pilotoinnissa rakennetaan kokonainen mahdollisimman lähelle lopullista järjestelmää vastaava testijärjestelmä. Pilottijärjestelmä myös otetaan käyttöön niin, että loppukäyttäjät käyttävät järjestelmää aidossa ympäristössä (Bansler & Havn 2010, 638). Osapilotit ja erityisesti minipilotit voivat sen sijaan olla korkeintaan erilaisia pienimuotoisia testejä jonkin ominaisuuden tai pienemmän kokonaisuuden toimivuuden testaamiseksi, koska muuten niiden vaatima työmäärä kasvaa nopeasti kohtuuttomaksi. Tällaisiin testeihin ei yleensä ole järkevää toteuttaa todellisessa ympäristössä toimivaa järjestelmää, kuten laajemmissa pilotoinneissa, vaan kysymys on enemmän palautteen keräämisestä.

4.2.3 Jatkuva testaaminen pilottiprojektien vaihtoehtona

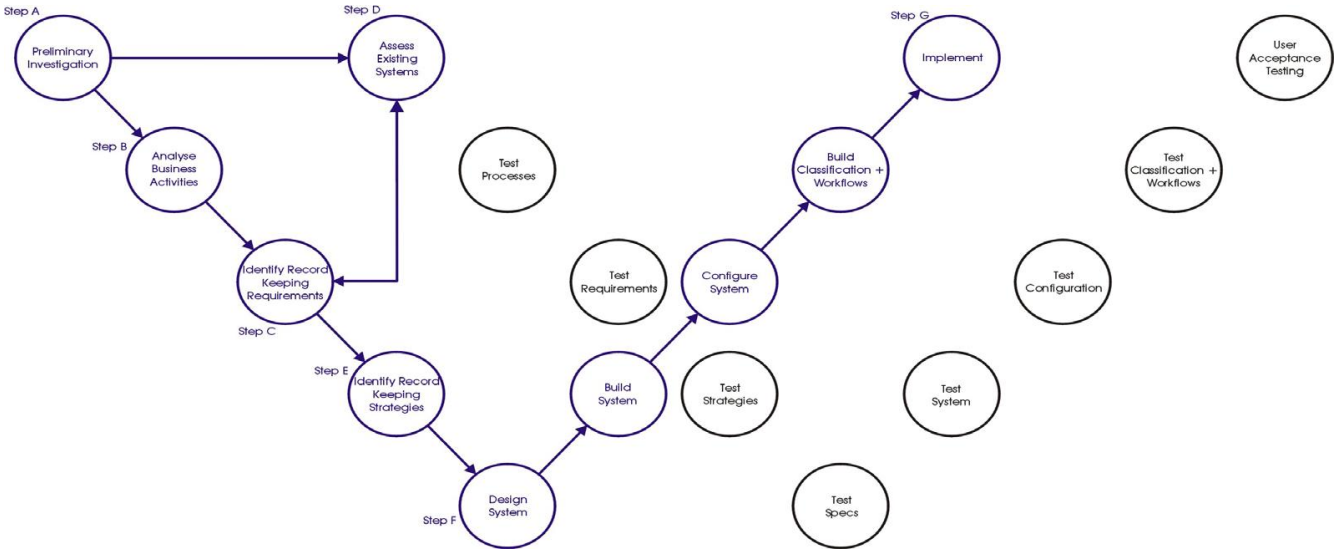
Koska pilottiprojektit ovat niin raskaita, voi kysyä, onko pilottiprojektien käyttäminen ollenkaan järkevä tapa kerätä palautetta pienemmistä suunnittelun kokonaisuuksista. Toinen vaihtoehtoinen hieman pilottiprojekteja muistuttava mutta kevyempi lähestymistapa pienempien kokonaisuuksien testaamiseen ja palautteen keräämiseen on jatkuva testaaminen. Jatkuvalla testaamisella tarkoitetaan suunnittelun tulosten testaamista niin, että jokaista suunnittelun vaihetta seuraa välittömästi kyseisen vaiheen tulosten testaaminen. Jatkuvan testaamisen perusasetelma kuvataan usein niin sanotulla W-mallilla, kuten tässä Gerrardin (2009) esityksessä:



Kuvio 15. Gerrardin W-malli ohjelmistojen suunnitteluun (Gerrard, 2009)

Gerrardin (2009) mallissa näkyy, kuinka testaaminen käynnistyy rinnakkain suunnittelun kanssa niin, että yksittäisiä suunnittelun vaiheita testataan aina suunnitteluvaiheen suorittamisen jälkeen ja testaus etenee suunnittelun kanssa rinnakkaisena prosessina. Näin voidaan tarkkailla suunnittelun etenemistä vaatimusmäärittelyiden mukaisesti ja havaita nopeasti suunnittelussa tapahtuneita virheitä ja puutteita. W-malli edistää tehokkaasti laadun toteutumista suunnittelussa. Laatu tarkoittaa tässä yhteydessä yleensä järjestelmän virheiden vähäisyyttä ja järjestelmän vastaavuutta käyttäjien toiveiden ja tarpeiden kanssa (Johnston 21, 31–32). Usein suunnittelussa käy niin, että suunnittelun edistyessä erkaannutaan koko ajan kauemmas siitä, mitä asiakkaat järjestelmältä odottavat ja mitkä heidän tarpeensa ovat. Jokainen vaihe vie järjestelmää kauemmas näistä tarpeista, jos suunnittelun täsmäämistä vaatimusten kanssa ei valvota ja näin järjestelmän laatu heikkenee. W-mallissa jatkuvat testaaminen varmistaa osaltaan laadun toteutumisen. (Roodenriis 2009)

Omaa työni kannalta mielenkiintoista on se, että Johnston (2004) on DIRKS:n vaihtoehtoista tulkintaa kuvaavassa suunnittelumallissaan käyttänyt pohjana jatkuvaa suunnittelua ja W-mallia. Johnstonin mallissa DIRKS:n suunnitteluvaiheita on muokattu niin, että ne ovat paremmin sovitettavissa W-malliin. Mallissa on lisätty DIRKS:n suunnitteluvaiheiden rinnalle niitä vastaavat testit:

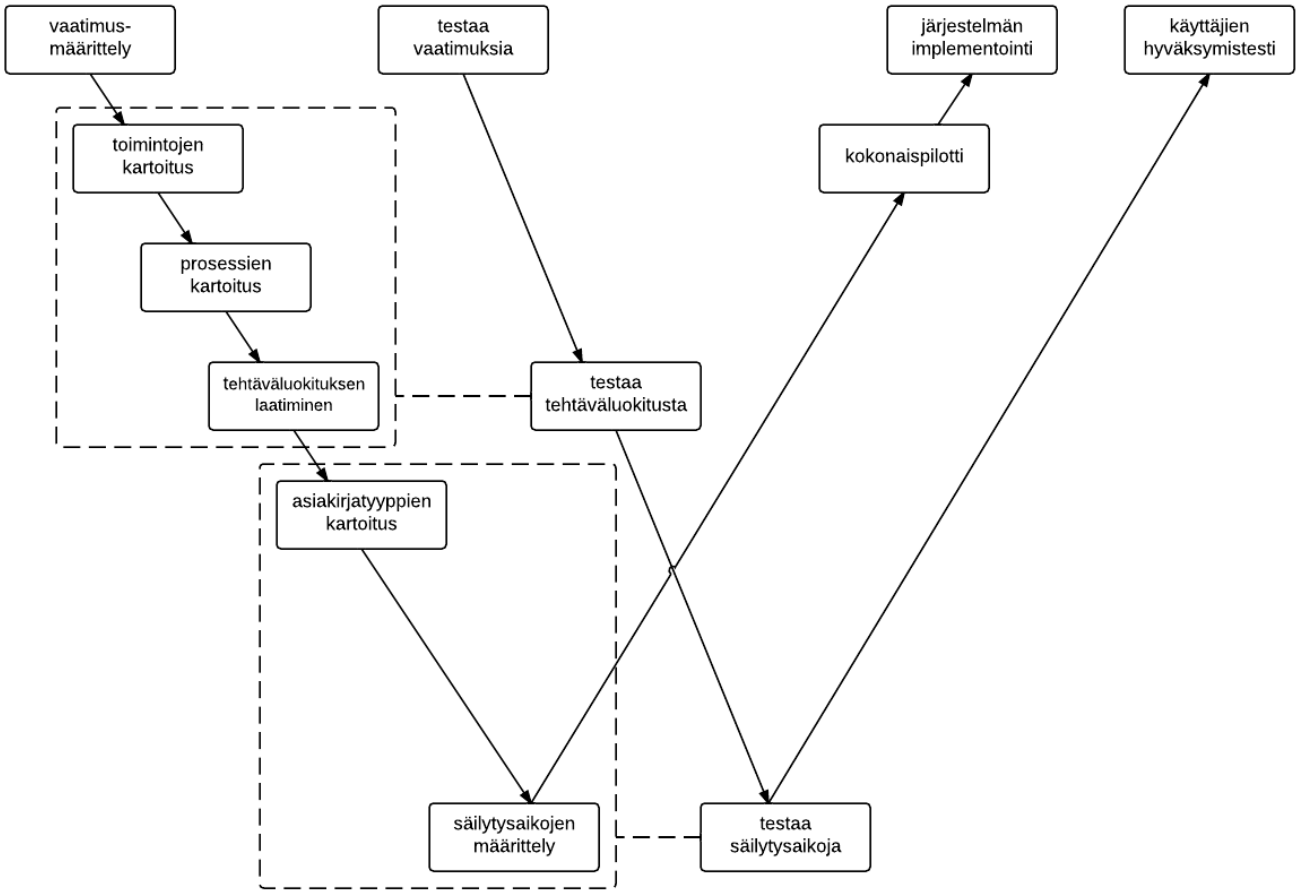


Kuvio 16. Johnstonin W-mallin pohjalta rakentama asiakirjajärjestelmän suunnittelumalli (Johnston 2004, 43)

Tämä Johnstonin malli on osittain päällekkäinen oman suunnittelumallini kanssa, sillä siinä on mallini tavoin kuvattu DIRKS:n pohjalta esimerkiksi tehtävien kuvaaminen, järjestelmän suunnittelu, johon myös tietosisällön suunnittelu sisältyy ja järjestelmän käyttöönotto. Koska malli kuvaa koko DIRKS:iä, se on kuitenkin DIRKS:n tavoin laajempi kuin oma vain tietosisällön suunnitteluun tarkoitettu mallini. Johnstonin mallin pohjalta on silti mahdollista hahmotella, miten jatkuva testaaminen voisi toimia omassa mallissani, jos mallini suunnitteluvaiheet piirrettäisiin W-malliin ja niiden rinnalle lisättäisiin jatkuva testaaminen:

Kuvio 17. Sähköisen asiakirjajärjestelmän tietosisällön suunnittelumalli W-malliin piirrettynä

Omassa mallissani oikea puoli jää lähes tyhjäksi, koska mallista puuttuvat kokonaan järjestelmän tekniseen toteuttamiseen liittyvät vaiheet. Vasemmalla puolella puolestaan on Johnstonin malliin verrattuna yksityiskohtaisempi jaottelu, koska sähköisen asiakirjajärjestelmän sisällön suunnittelu on tässä jaettu osiin. Johnstonin ja Gerrardin malleissa suunnittelu on vain yksi vaihe järjestelmän kehittämistä. Omassa mallissani testauksen vaiheet menevät yksityiskohtaisemmalle tasolle ja voi kysyä, onko testaamisen järjestäminen enää mielekästä näin pienille yksittäisille suunnittelun vaiheille. Mikään ei tietysti estä testaamasta, vastaako esimerkiksi asiakirjatyyppien kartoitus tavoitettaan, mutta tällä tavalla testaaminen jakaantuisi varsin pieniin osiin. Olisi myös mahdollista yhdistää yksittäisiä vaiheita ja asettaa testauksia vain isommille kokonaisuuksille. Omassa mallissani luontevia kokonaisuuksia ovat osittamisen yhteydessä esitellyt osakokonaisuudet niin, että tehtäväluokitusta testattaisiin vain yhden kerran, kun sekä toimintojen että prosessien kartoituksen sisältävät osakokonaisuudet ovat valmiita. Näin testattavia tuloksia olisi vain kaksi: tehtäväluokitus ja säilytysajat:



Kuvio 18. W-malli osakokonaisuuksiin sovellettuna. Kokonaisuuden osat yhdistetty katkoviivalla.

Tällä tavalla toteutettuna testaaminen yksinkertaistuu ja erillisiä testauksia on vähemmän. Voi kuitenkin kysyä, mitä testaamisella saavutetaan verrattuna pilottiprojekteihin. Testaaminen

todennäköisesti nostaa järjestelmän laatua, koska testaaminen keskeinen idea on tarkkailla ja testata sitä, että järjestelmän suunnittelu vastaa eri vaiheissaan järjestelmälle asettuja vaatimuksia. Omassa mallissani olen kuitenkin asettanut eri ratkaisujen valitsemisen kriteereiksi nopeuden ja joustavuuden. Testaamisella voidaan saavuttaa laatua, mutta nopeutta se ei tuo ja myöskään yhteys järjestelmän käyttäjiin ei lisäänny, jos testaus toteutetaan lähinnä vertailuna järjestelmävaatimuksiin. Tietysti testaukseen voidaan tuoda mukaan myös käyttäjiä ja kerätä heiltä palautetta toteutuksesta, mutta sitten testaus alkaa muistuttaa paljolti pilottiprojektia.

Prototyypit ja pilottiprojektit vaikuttavat selkeämmiltä yhdistää pelkkää nopeutta ja joustavuutta korostavaan malliin, koska niitä voidaan toteuttaa tarpeen mukaan ja erilaajuisina. Jatkuvaa arviointia tunnutaan yleisestikin sovellettavan esimerkiksi Johnstonin (2004, 43) ja Remenyin ja Sherwood-Smithin (1999, 20) mallien perusteella laajemmalla tasolla kuin omassa mallissani kuvattu tietosisällön suunnittelu. Voisi olla perusteltua soveltaa jatkuvaa arviointia mallissani korkeintaan kerran koko suunnittelun laajuisessa iteraatiossa esimerkiksi jokaisen osakokonaisuuden sijaan.

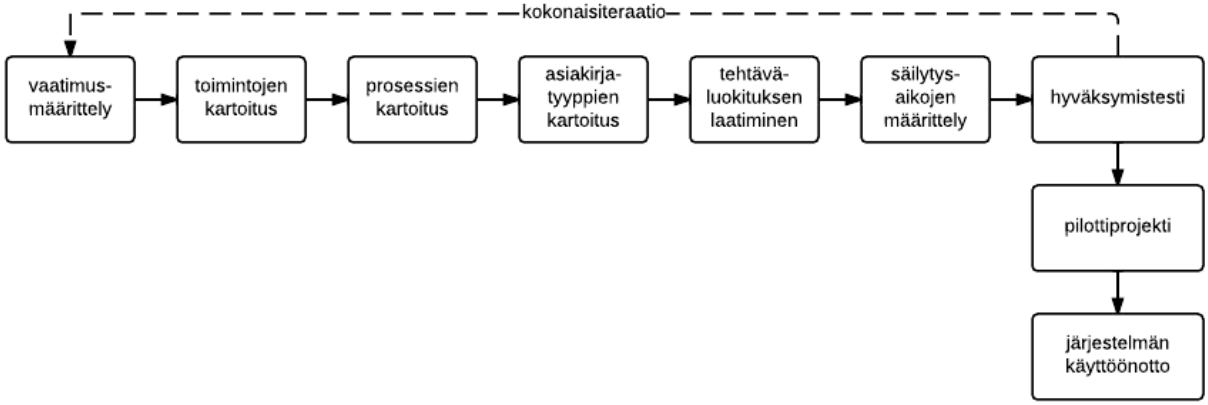
Toisaalta vaikka arviointi ei välttämättä edistä mallin nopeutta, Kautzin (2011, 228) mielestä testaaminen on yksi tapa tuoda käyttäjiä mukaan suunnitteluun, kerätä heiltä palautetta ja antaa heille valtaa suunnittelun päätöksiin. Tämän takia jatkuvan arvioinnin kaltainen palautteenkeräysmekanismi tuntuisi perustellulta myös omassa mallissani, jossa pyrin korostamaan käyttäjiltä saatua palautetta. Kautzin (emt.) tutkimassa projektissa palautteen keräämisen keskeinen vaihe oli aiemmin esitellyn Gerrardin mallin tavoin niin sanottu hyväksymistesti koko järjestelmän iteraatioiden välillä. Tässä koko järjestelmän tasolla tapahtuvassa testissä käyttäjillä on mahdollisuus kokeilla järjestelmää kokonaisuudessaan ja sen pohjalta vaikuttaa suunnitteluun seuraavassa iteraatiossa.

Käyttäjille annetaan hyväksymistesteissä enemmän valtaa kuin suunnittelijalähtöisessä käyttäjätestaamisessa. Ajatuksena on, että käyttäjät ja mahdollisesti muut projektiin osallistujat arvioivat sitä, toimiiko järjestelmä toivotulla tavalla ja osallistuvat aktiivisesti vaatimusten kehittämiseen ja suunnittelun suuntaa voidaan arvioinnin pohjalta muuttaa radikaalistikin tai jopa lopettaa se. (Collyer et al. 2010, 109) Hyvin toteutetun testaamisen idea yleensäkin on se, että käyttäjille annetaan aidosti mahdollisuus vaikuttaa järjestelmän suunnitteluun ja sen vaatimuksiin. Tämä vaatii suunnittelijoilta joustavuutta, koska järjestelmän vaatimukset voivat muuttua jatkuvasti suunnittelun edetessä. Tämän takia käyttäjälähtöinen arviointi sopii hyvin

yhteen ketterän kehityksen menetelmien kanssa, koska ketterän kehityksen menetelmät mahdollistavat arviointien pohjalta suunnitteluun tehtävien muutosten toteuttamisen joustavasti. (Remenyi & Sherwood-Smith 1999, 27, 29)

Käyttäjien vaikutusmahdollisuuksien lisäämisen ohella koko järjestelmän laajuinen kerran iteraatiokierroksessa tapahtuva testaaminen ja vaatimusten päivittäminen ovat hyviä keinoja varmistaa, että järjestelmän laatu pysyy korkeana. Remenyin ja Sherwood-Smithin (1999, 14) mukaan jatkuvalla testaamisella pyritään varmistamaan laadukkaiden informaatiojärjestelmien suunnittelu ja se, että järjestelmät oikeasti tuottavat organisaatiolle suoria liiketoimintaa tukevia hyötyjä. Myös yksittäiset testaukset tukenevat tätä tavoitetta jossain määrin, vaikka eivät niin tehokkaita olekaan kuin jatkuva testaaminen.

Omassa mallissani suunnittelun nopeuden lisäämiseksi tällaisten kevyempien koko järjestelmän suunnittelun iteraatioiden välissä toteutettavien hyväksymistestien toteuttaminen voisi olla hyvä vaihtoehto laajempien pilottiprojektien hyödyntämiselle. Vaativuutensa takia voisi olla järkevää toteuttaa pilottiprojekti vasta ennen järjestelmän käyttöönottoa ja sitä ennen käyttää testaamiseen tällaisia nopeampia hyväksymistestejä. Tällä tavalla toteutettuna malli näyttäisi tältä:



Kuvio 19. Hyväksymistestin hyödyntäminen kokonaisiteraatioiden testaamisessa ja palautteen keräämisessä

Tällainen lähestymistapa paitsi edistää asiakkaiden palautteen keräämistä, tukee myös Remenyin ja Sherwood-Smithin (1999, 15, 29) mukaan hyvin iteratiivista kehittämismallia ja sopii hyvin yhteen prototyyppien ja pilottiprojektien hyödyntämisen kanssa. Arvioinnin toteuttaminen tuottaa tietoa tavoitteiden täyttymisestä ja uudelleenmäärittelytarpeesta, jotka voidaan sitten siirtää järjestelmän kehityksen seuraavaan iteraatioon. Näin arviointi tukee järjestelmän suunnittelussa prototyyppien rakentamista palautteenkeräämismekanismina ja mah-

dollistaa järjestelmällisemmän palautteen keräämisen ja suunnittelun arvioinnin kokonaisiteraation lopussa ja tukee seuraavan iteraation vaatimusmäärittelyiden toteuttamista.

Puutteena lähestymistavassa voi nähdä sen, että testaamisen pitäisi optimaalisesti olla jatkuvaa, että pystyttäisiin tehokkaammin ja joustavammin varmistamaan, että suunnittelu etenee asetettujen tavoitteiden mukaisesti (Remenyi & Sherwood-Smith 1999, 18). Suunnittelun nopeuden ja joustavuuden kannalta jatkuva testaaminen ei kuitenkaan ole mallissani perusteltua. Jos laadunhallintaa halutaan kuitenkin suunnittelussa korostaa, jatkuva testaaminen on todennäköisesti kohtuullisella vaivalla yhdistettävissä malliin.

Remenyin ja Sherwood-Smithin (1999, 19) mukaan koko projektin lopussa tehtävistä arvioinneista on usein vain vähän hyötyä kehittäjien oppimisen kannalta ja testaamista pitäisi tehdä siksi toistuvasti. Malliin lisäämäni lähestymistapa testaamisesta on kompromissi jatkuvan testaamisen ja yhden projektin lopussa tapahtuvan testaamisen välillä. Vaikka tämä lähestymistapa ei ole optimaalinen, uskon, että se on hyödyllinen projektin hallittavuuden kannalta, koska se tuottaa aina iteraatioiden välillä tietoa seuraavalla iteraatiolle ja toistuu useammin suunnittelun aikana kuin pelkän loppuarvioinnin toteuttaminen. Arvioinnin hyöty on kuitenkin riippuvaista paitsi sen määrästä myös laadusta. Remenyin ja Sherwood-Smithin (emt.) mukaan testaamisen määrän puutteiden lisäksi ongelmana on myös se, että arviointi ei ole riittävän perusteellista ja tarkkaan määriteltyä. Vaikka testaus suoritettaisiin vain kerran kokonaisiteraatiossa, jos se tehdään huolellisesti ja järjestelmällisesti, se voi tuottaa arvokasta tietoa suunnittelun etenemisestä.

4.2.4 Prototyypit

Jatkuvan testaamisen lisäksi palautteen keräämistä suunnittelun kokonaisuuksista voidaan tehostaa myös prototyyppien käytöllä. Pilottiprojektien yhteydessä käsiteltiin niin sanottujen minipilottien hyödyntämistä. Bansler & Havn (2010, 647) ovat kuitenkin todenneet, että jos pilottiprojektien mittakaavan määrittely on vaikeaa, eikä pilotin toteuttaminen tunnu mielekkäältä projektin testaamiseen, voi olla pilottien vaativuuden vuoksi järkevämpää jättää ne toteuttamatta ja luottaa enemmän esimerkiksi prototyyppien hyödyntämiseen ja suunnittelun jakamiseen pienempiin osiin.

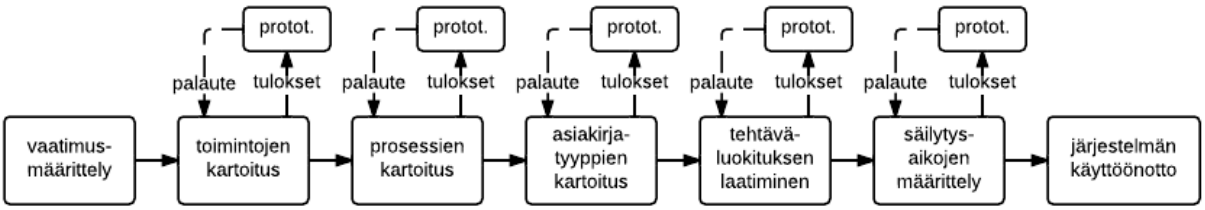
Prototyyppi on tuotesuunnittelusta ja tietojenkäsittelytieteestä ketterään kehitykseen lainattu termi. Järjestelmäsuunnittelussa prototyyppi tarkoittaa sen hetkistä valmista järjestelmää imitoivaa järjestelmää, jonka avulla voidaan nähdä konkreettisesti millainen järjestelmä on ja testata sitä (Center for Technology in Government 1998, 5). Ketterässä kehityksessä prototyypit ovat tärkeitä, koska käyttäjien merkitystä suunnittelussa korostetaan ja järjestelmää pyritään kehittämään käyttäjiltä saatavan palautteen perusteella paremmaksi. Evansin ja Rouchen (2006, 320–328) mukaan prototyyppien rakentaminen varmistaa sen, että kaikki kehitysprosessiin osallistuvat voivat oppia järjestelmän ongelmista ja ratkaisuista ja pystyvät näkemään konkreettisesti suunnitteluratkaisujen vaikutuksen järjestelmään. Tämä käyttäjäkeskeisyys on tärkeimpiä syitä prototyyppien korostamiseen ketterässä kehittämisessä.

Myös lineaarisissa suunnittelumalleissa korostetaan usein käyttäjäpalautteen merkitystä järjestelmien kehittämisessä, mutta käytännössä kattavan palautteen saaminen ilman, että käyttäjät pääsevät konkreettisesti kokeilemaan järjestelmää ja ymmärtävät sen vaikutukset työskentelyynsä, on usein vaikeaa tai melkein mahdotonta. (Bansler & Havn 2010, 638) Esimerkiksi Boehmin, Grayn ja Seewaldtin (1984) mukaan prototyyppien hyödyntäminen on menetelmänä melkein kaksi kertaa niin tehokas kuin tarkkojen vaatimusmäärittelyjen laatiminen projektin alkuvaiheessa.

Pilottiprojektit ja prototyypit ovat käsitteellisesti lähellä toisiaan, mutta niitä käytetään hieman eri tarkoituksiin. Prototyypit ovat nopeasti rakennettuja, oikeaa järjestelmää mallintavia järjestelmiä, jotka voidaan hylätä nopeasti ja joiden tarkoituksena on usein vain esitellä jotain järjestelmän osaa tai toimintoa palautteen keräämiseksi. Prototyyppien vahvuutena on niiden joustavuus, mutta ongelma on usein se, että niitä testataan kontrolloidussa ympäristössä, joka ei vastaa käyttäjien arkiympäristöä haasteineen. Verrattuna pilottiprojekteihin prototyypit ovat helpompia rakentaa ja joustavampia, mutta pilottiprojekteista saadaan yleensä totuudenmukaisempi ja kattavampi kuva järjestelmän ongelmista. (Bansler & Havn 2010, 638–639) Rzevskin (1984) mukaan pilottijärjestelmät ovat huolellisesti suunniteltuja ja toimivia järjestelmiä, joita voidaan kehittää suoraan valmiiksi järjestelmiksi (emt.) Erilaisiin ja eritasoisiin suunnittelun vaiheisiin on usein järkevää soveltaa joko prototyyppejä tai pilottiprojekteja riippuen siitä, kuinka nopeasti palautetta halutaan ja kuinka laajasta kokonaisuudesta on kysymys.

Sähköisen asiakirjajärjestelmän tietosisällön suunnittelun yhteydessä prototyyppitermin käyttäminen voi olla hämäävää, koska prototyypit rinnastetaan yleensä erilaisten tuotteiden tai ohjelmistojen suunnitteluun. Käytän kuitenkin termiä prototyyppi, koska se on ketterässä kehittämisessä yleisesti käytetty termi ja lisäksi prototyyppi kuvaa hyvin sitä, että luodaan suunnittelukokonaisuutta konkreettisesti kuvaava esitys, josta käyttäjät voivat paremmin hahmottaa, miten suunnittelukokonaisuus toimii. Prototyyppejä ei tässä yhteydessä pidä ajatella liian jäykästi vain tuotetta tai järjestelmää mallintavina esityksinä, vaan ne voivat olla esimerkiksi luokituksen visualisointeja paperilla, tietosisällön esittelyä vanhassa järjestelmässä tai tietosisällön toiminnan esittelyä muuten järjestelmän osana (Center for Technology in Government 1998, 5). Prototyyppien hyödyntämisessä keskeistä ei ole toteutustapa, vaan se miten hyvin se pystyy havainnollistamaan käyttäjälle suunnittelun tuloksen, josta suunnittelijat haluavat palautetta. Mitä paremman kuvan käyttäjät saavat suunnittelun tuloksen toiminnasta, sitä paremmin prototyyppi toimii.

Prototyyppejä voidaan soveltaa luontevasti suunnittelun yksittäisiin vaiheisiin niin, että prototyyppien avulla tehostetaan palautteen keräämistä iteraatioiden välillä. Käytännössä tämä toimii niin, että yksittäisen vaiheen, esimerkiksi toimintojen kartoituksen, pohjalta laaditaan prototyyppi, jonka avulla kerätään palautetta saman vaiheen seuraavaa iteraatiota varten:



Kuvio 20. Prototyyppien käyttäminen yksittäisten vaiheiden palautteen keräämiseen ja iterointiin

Tällä tavalla toteutettuna prototyypit korvaavat minipilotit yksittäisten vaiheiden testaamisessa ja kehittämisessä. Prototyypit ovat luontevampi valinta näin matalalla suunnittelun tasolla, koska verrattuna pilottiprojektien toteuttamiseen niitä voidaan kehittää pienemmillä resursseilla ja joustavammin. Myös osakokonaisuuksien testaamisessa ja iteroinnissa prototyypit ovat joustavuutensa takia todennäköisesti yleensä parempi valinta kuin osapilotit.

Sen lisäksi, että prototyypeillä testataan perussuunnittelua, niitä voidaan hyödyntää myös niin, että niiden avulla testataan kilpailevia suunnittelutoteutuksia. Collyerin et. al (2010, 109) mukaan on usein järkevää kokeilla käytännössä mahdollisimman varhaisessa vaiheessa sitä, miten erilaiset vaihtoehtoiset suunnitteluratkaisut toimisivat käytännössä ja valita näistä opti-

maalisin lähestymistapa järjestelmän toteutukseen. Nämä kilpailevat suunnitteluratkaisut voivat olla esimerkiksi suunnittelijoiden ideoita toteutuksesta tai käyttäjiltä saatuja lupaavia ehdotuksia. Prototyypit ovat hyvä tapa testata kilpailevia ideoita ja saada tietoa nopeasti siitä, miten ne käytännössä toimivat.

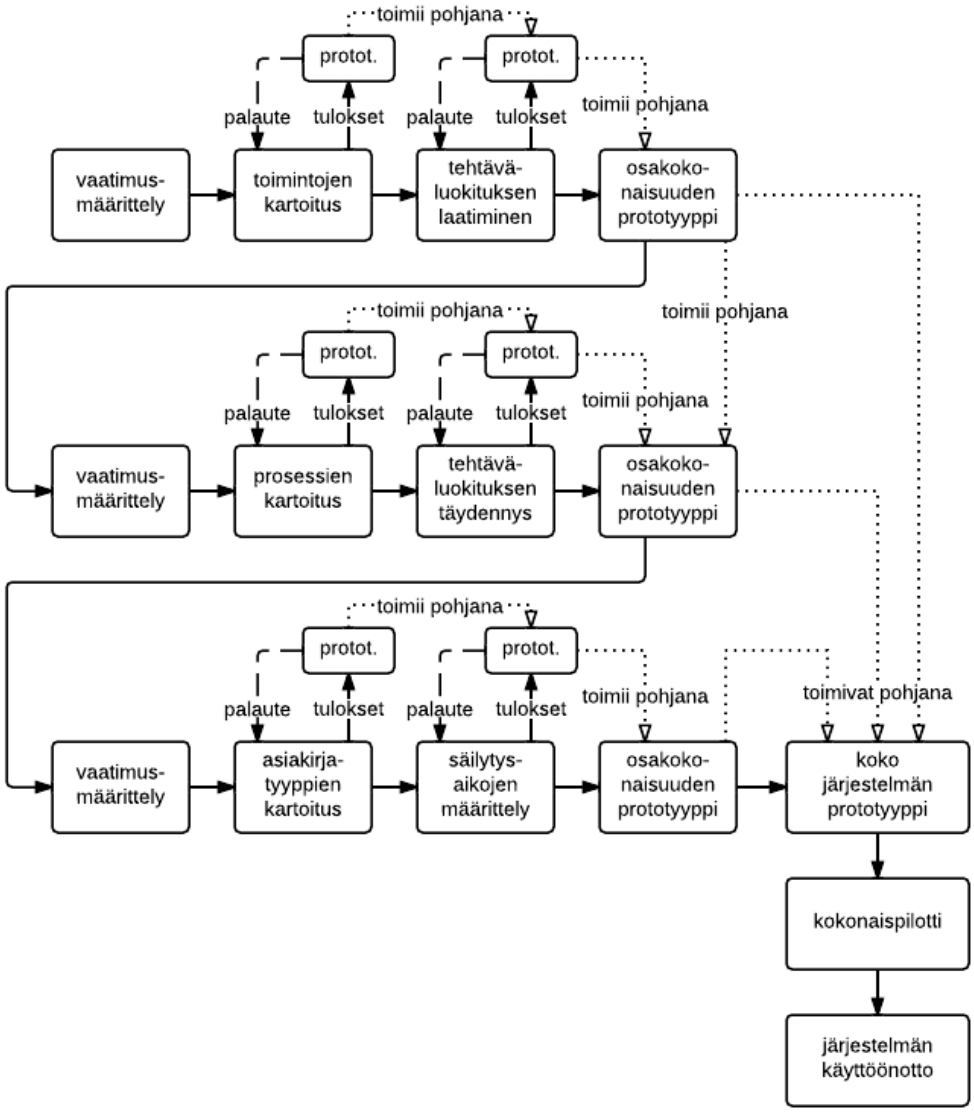
Vaikka nopeasti ajatellen tällaisten vaihtoehtoisten ratkaisujen testaaminen vie paljon aikaa, se on Collyerin et al. (2010, 117) mukaan silti yleensä järkevää, koska näin voidaan testata erilaisia lähestymistapoja ja hylätä mahdollisimman nopeasti toimimattomat ratkaisut. Joustavuuden kannalta tällainen lähestymistapa vaikuttaa myös perustellulta, sillä näin voidaan testata tehokkaasti uusia ideoita toteuttamatta laajempia projekteja.

Prototyyppeihin pätevät tosin jo pilottiprojektien yhteydessä esitetyt ongelmat, eli niiden toteutus vie resursseja ja hidastaa välitavoitteiden saavuttamista ja kokonaisten suunnittelun iteraatioiden valmistumista. Toisaalta prototyypit ovat niin tehokas keino palautteen keräämiseen käyttäjiltä, että niiden toteuttaminen on usein perusteltua, jos ei voida olla suhteellisen varmoja, että tehdyt suunnitteluratkaisut ovat käyttäjien kannalta tyydyttäviä. Tällainen riittävä varmuus käyttäjien tarpeista voisi toteutua esimerkiksi silloin, jos suunnittelu on pilkottu organisaation osittain ja aiemman osan suunnittelusta saatuja kokemuksia voidaan soveltaa toiseen samankaltaiseen osaan.

Haasteina prototyyppien käyttämisessä ovat niiden vaatimat resurssit ja se, että niiden hyödyntäminen vaatii erilaista osaamista verrattuna muuhun projektin toteuttamiseen. Prototyyppien rakentaminen on myös haastavaa laajojen informaatiojärjestelmien suunnittelussa, koska on usein vaikeaa määritellä, millä perusteilla järjestelmiä pitäisi jakaa osiin prototyyppien rakentamista varten. (Alavi 1984, 558)

Voi myös kysyä, miten prototyypit pitäisi käytännössä toteuttaa. Tehdäänkö jokaiselle vaiheelle oma uusi prototyyppinsä vai kehitetäänkö järjestelmää askel askeleelta kohti parempaa lopputulosta ja rakennetaan uusia prototyyppejä vanhojen päälle. Ohjelmistosuunnittelussa prototyyppejä rakennetaan usein niin, että vanhan prototyypin pohjalta kehitetään seuraavia versioita prototyypeistä (ks. esim. Najjar 1990). Päällekkäin rakentaminen on kuitenkin sähköisen asiakirjajärjestelmän tietosisällön suunnittelussa vaikeampaa, koska tietosisällön suunnittelu jakaantuu toisistaan pitkälti erillisiin osakokonaisuuksiin. Toki esimerkiksi prosessien kartoittamisella ja säilytysaikojen määrittelyllä on looginen yhteys. Säilytysaikojen määrittely

seuraa suunnittelussa loogisesti prosessien kartoittamista. Nämä kokonaisuudet ovat kuitenkin enimmäkseen itsenäisiä ja toisistaan erillisiä suunnittelukokonaisuuksia. Kuviossa 21 olen hahmotellut, miten aiemmin kehitettyjä prototyyppejä voitaisiin käyttää toisten prototyyppien pohjana, jos prototyyppejä rakennettaisiin ohjelmistosuunnittelun tavoin toistensa päälle. Kuviossa on piirretty näkyviin myös osakokonaisuudet ja niiden testaamiseen käytetyt prototyypit ja se, miten näitä osakokonaisuuksien prototyyppejä voitaisiin käyttää koko järjestelmän prototyypin pohjana:

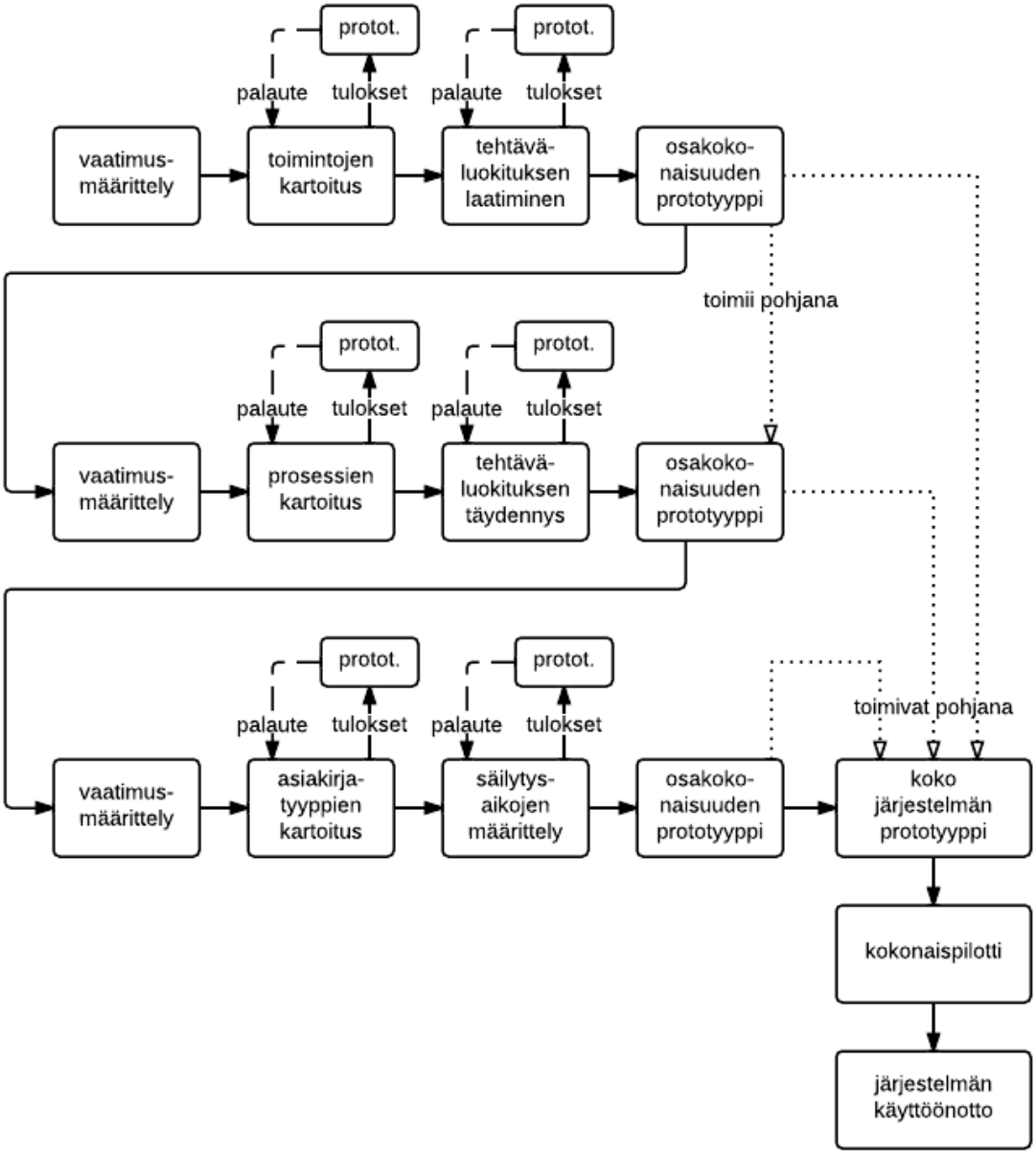


Kuvio 21. Prototyyppien rakentuminen toistensa päälle

Kuten kuvasta näkyy, järjestelmä menee kuitenkin monimutkaiseksi tällä tavalla ajateltuna, vaikka kuvaan ei ole edes piirretty eri osakokonaisuuksien yksittäisten vaiheiden välisiä yhteyksiä, vaikka niitäkin voisi olla. Esimerkiksi toimintojen kartoituksessa käytetty prototyyppi voisi toimia toteutustavaltaan prosessien kartoituksen pohjana ja ensimmäisen osakokonai-

suuden tehtäväluokituksen laatiminen toisen vaiheen tehtäväluokituksen vaiheen pohjana. Käytännössä eri vaiheissa toteutettavat prototyypit ovat erilaisia tarpeen mukaan ja niiden rakentaminen toistensa päälle ei usein toimi. Tämän takia suunnitteluvaiheita koskevia prototyyppejä ei kannata yleensä rakentaa järjestelmällisesti toistensa päälle. Yksittäisten vaiheiden prototyyppien hyödyntäminen muiden prototyyppien suunnittelussa on tapauskohtaista. On luontevampaa käyttää prototyyppejä pääosin uudestaan yksittäisen vaiheen sisällä niin, että samaa prototyyppiä, jota on käytetty aiemmissa iteraatioissa samassa vaiheessa, hyödynnetään suoraan pohjana uudessa prototyypissä, jos tarpeet testaamiselle eivät ole merkittävästi muuttuneet.

Osakokonaisuuksien osalta poikkeus prototyypin hyödyntämisessä toisen osakokonaisuuden prototyypin pohjana voisi olla tehtäväluokituksen laatiminen, joka jakaantuu mallissa kahteen osaan. Toinen osakokonaisuus lähinnä täydentää ensimmäisessä osakokonaisuudessa luotua tehtäväluokitusta prosessien kartoittamisen pohjalta. Tämän takia ensimmäisen osakokonaisuuden prototyyppi lienee usein samankaltainen toisen osakokonaisuuden kanssa. Lisäksi yksittäisten osakokonaisuuksien prototyyppien hyödyntäminen koko järjestelmän prototyypin rakentamisessa voi olla toimiva ratkaisu, jos osakokonaisuuksien prototyypit ovat tekniseltä toteutukseltaan samankaltaisia koko järjestelmän prototyypin kanssa.



Kuvio 22. Karsittu prototyyppien rakentuminen toistensa päälle

Prototyyppien rakentaminen toistensa päälle on mahdollisuuksien mukaan hyödyllistä, koska se nopeuttaa suunnittelua ja lisää myös joustavuutta, koska käyttäjiltä saadun palautteen kerääminen nopeutuu. Korostettaessa suunnittelun nopeutta ja joustavuutta kannattaa pääsääntöisesti käyttää kaikkia mahdollisia aiempia kokemuksia ja suunnittelun välineitä uudestaan niin paljon kuin mahdollista niin kauan kuin tämä ei tarkoita sitä, että uusia lähestymistapoja ei kokeilla.

Prototyyppien ja muiden palautteenkeräämisvälineiden osalta kannattaa huomata se, että ne eivät tietysti ole ainoa tapa kerätä palautetta suunnittelusta. Usein prototyypin voi suunnittelussa korvata aktiivisella kanssakäymisellä käyttäjien kanssa. Esimerkiksi Kautzin (2011,

232) tutkimuksessaan esittelemässä onnistuneessa suunnitteluprojektissa monia suunniteltuja tiedonkeräämisen menetelmiä ei käytännössä ehditty toteuttamaan, mutta tiivis yhteistyö ja palautteen kerääminen keskustelemalla käyttäjien kanssa virallisempien laajempien kokonaisuuksien testien ohella korvasivat apuvälineiden puutteet.

4.2.5 Iterointi

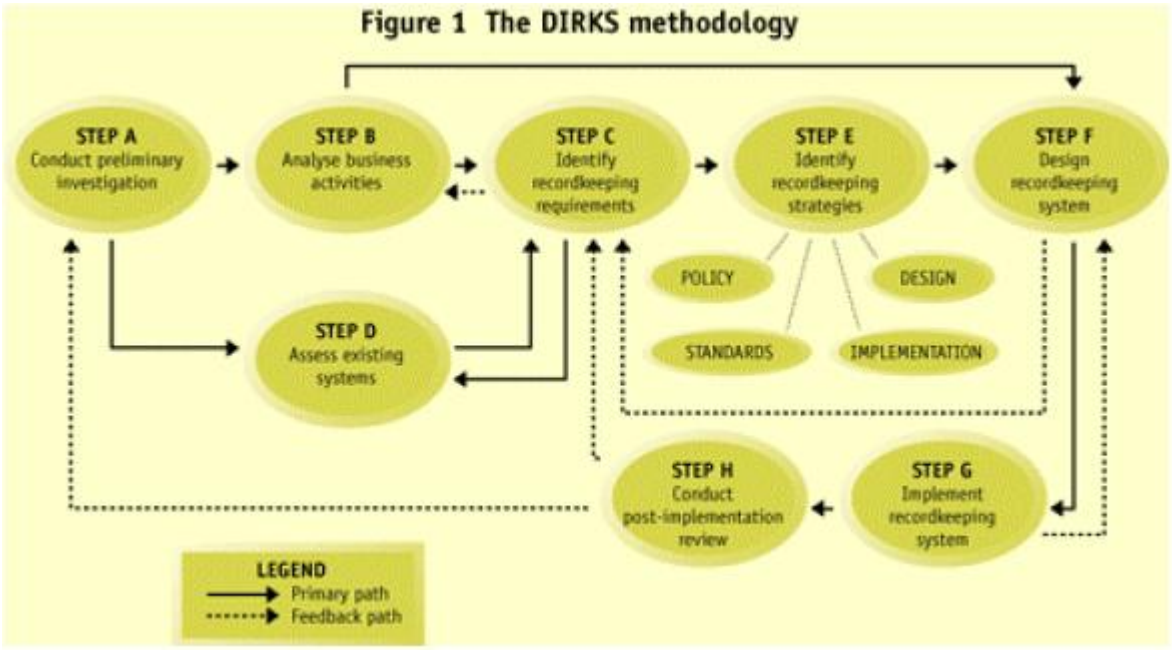
Iterointi on erityisesti matematiikassa ja tietojenkäsittelytieteessä käytetty termi, jolla tarkoitetaan tulokseen pyrkimistä toistamalla samankaltaista toimenpideketjua kerta toisensa jälkeen niin, että koko ajan lähestytään haluttua lopputulosta. Iteroinnilla ei yleensä pyritä täsmälliseen lopputulokseen, vaan pääsemään tarpeeksi lähelle haluttua tulosta. Iteraatio puolestaan viittaa yhteen toimenpideketjun kierrokseen, joiden lopputuloksena saadaan tarkempi välitulos. Ketterien kehitysmenetelmien yhteydessä käytetään yleisesti iterointi-termiä kuvaamaan sitä, miten järjestelmän suunnittelua suoritetaan peräkkäisinä suunnilleen samankaltaisina toimenpideketjuina, joilla järjestelmästä tehdään kerta kerralta enemmän vaatimusten mukainen ja kehitetään vaatimuksia saadun palautteen pohjalta iteraatioiden välillä. (Center for Technology in Government 1998, 4)

Iterointia hyödynnetään informaatiojärjestelmien suunnittelussa usein niin, että pyritään julkaisemaan mahdollisimman nopeasti pienimmän mahdollisen mittakaavan välimalli, josta voidaan kerätä käyttäjiltä palautetta ja sen jälkeen hiljalleen kehittää järjestelmää paremmaksi ja täydellisemmäksi (Collyer 2010, 109). Aiemmin esitelty osittaminen ja suunnittelun laajentaminen hiljalleen laajempiin organisaation osiin ja suunnittelun kokonaisuuksiin tukee osittain tätä tavoitetta, mutta iterointi on ehkä keskeisin vaiheittaisen julkaisemisen toteutusväline. Iteroimalla voidaan vähentää suunnittelun riskejä ja kehittää nopeasti konkreettisia välineitä käyttäjille järjestelmän arvioimiseen. Virheiden havaitseminen varhaisessa vaiheessa auttaa merkittävästi laskemaan virheiden kustannuksia ja hylkäämään joustavammin epäonnistuneet kokeilut tai epäonnistuneen projektin mahdollisimman varhaisessa vaiheessa (emt.).

DIRKS-ohjeistuksessa iteratiivisuutta on hyödynnetty jonkin verran ja sen käyttö tuodaan esille sekä laajempien kokonaisuuksien välillä että yksittäisten vaiheiden suorittamisessa. Yksittäisten vaiheiden osalta DIRKS:ssä otetaan kantaa liiketoimintojen kartoittamiseen ja tehtäväluokituksen laatimiseen. DIRKS:n mukaan liiketoimintojen analysointi on iteratiivista toi-

mintaa, jossa lopullinen luokitus muodostuu usean suunnittelukierroksen tuloksena. Myös tehtävien kartoitusta ja tehtäväluokituksen laatimista suositellaan DIRKS:ssä tehtäväksi iteratiivisesti. Kartoitusprosessi nähdään kerroksittain ylhäältä alas etenevänä prosessina, jossa ensin yleisellä tasolla kartoitetaan laajemmat tehtäväkokonaisuudet ja siitä edetään koko ajan tarkempaan näkemykseen tehtävistä ja lopulta yksittäisten prosessien kuvaamiseen. Jokaiselle näistä tasoista voidaan palata ja niitä voidaan ohjeistuksen mukaan suorittaa useaan kertaan, että saavutetaan haluttu tulos (National Archives of Australia 2001, 6-9, osa 2B).

Iterointi on huomioitu DIRKS:ssä myös yleisemmällä tasolla. Alla olevassa kuvassa on esitetty DIRKS:n suunnittelukokonaisuus, jossa palautesilmukat näkyvät katkoviivoina:



Kuvio 23. DIRKS:n vaiheet ja palautesilmukat (National Archives of Australia 2001, 13, a user's guide)

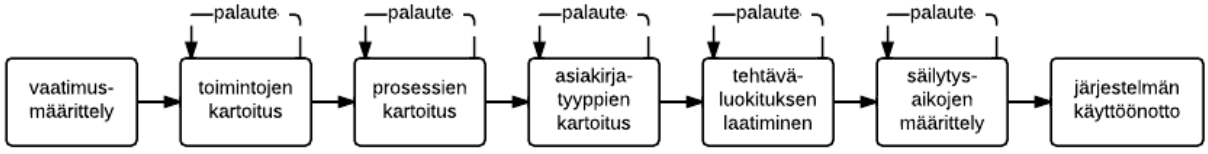
Tästä kuviosta näkyy kuinka DIRKS:ssä mallin suorittaminen on pyritty luomaan joustavaksi niin, että suunnittelu voidaan paitsi suorittaa erilaajuisena erilaisissa tilanteissa jättäen osa vaiheista väliin myös niin, että useiden vaiheiden välillä voidaan palata edellisiin vaiheisiin tarkentamaan mallia palautteen pohjalta. Kuviossa katkoviivanuolet kuvaavat erilaisia mahdollisia palautepolkuja ja yhtenäiset nuolet suunnitteluprosessin etenemistä vaiheesta toiseen.

DIRKS:n johdannon mukaan malli tosin perustuisi vesiputousmalliin, jossa edetään vaiheesta toiseen ja suoritetaan jokainen vaihe erillisenä toisistaan. Vaikka mallia DIRKS:n johdannossa kutsutaan vesiputoukseksi, se vaikuttaa enemmän muokatulta vesiputousmallilta, jossa voidaan tarvittaessa palata suunnittelussa taaksepäin suorittamaan aiempia suunnittelun vai-

heita (Jainin ja Chandrasekaranin 2009). DIRKS:ssä on palautesilmukoita, joiden kautta suunnittelussa voidaan palata aiempiin vaiheisiin ja näin iteratiivisesti parantaa järjestelmää. Myös DIRKS:n johdannossa puhutaan siitä, miten malli on syklinen ja siinä on iteratiivisuutta. Syklisyys tarkoittaa kuitenkin tässä tapauksessa sitä, että asiakirjajärjestelmää voidaan jatkuvasti parantaa ja suunnitella uudestaan käynnistämällä uusia suunnitteluprojekteja, kun järjestelmä osoittautuu puutteelliseksi. (National Archives of Australia 2001, 14, a user's guide)

Esimerkiksi palautekierros koko suunnittelukierroksen ympäri, eli se että viimeisestä vaiheesta H voidaan jälleen palata vaiheeseen A, tarkoittaa DIRKS:ssä enemmän sitä, että tarpeen mukaan suunnitteluprosessi voidaan käynnistää kokonaan uudelleen, kun siihen tulee tarvetta. Johnstonkin (2005, 3) on esittänyt, että DIRKS-mallin iteratiivisuus ei vastaa oikeasti ketterän kehityksen menetelmissä yleensä tarkoitettua iteratiivisuutta. DIRKS:n palautesilmukat on tarkoitettu kerran käytettäviksi sen sijaan, että ne muodostaisivat jatkuvan iteratiivisen prossin suunnittelussa. Omassa mallissani pyrin tällaiseen jatkuvaan iterointiin järjestelmän tietosisällön suunnittelemiseksi asteittain toistensa päälle rakentuvien vaiheiden avulla. DIRKS:n palautesilmukat toimivat kuitenkin hyvänä lähtökohtana sille, miten iteraatioita voi soveltaa.

Suunnittelumalliin voidaan lisätä DIRKS:n pohjalta yksittäisiin vaiheisiin, eli toimintojen ja prosessien kartoittamiseen ja tehtäväluokituksen laatimiseen, vaiheiden sisäinen iterointi. Kuviossa 24 on esitetty, kuinka iterointi yksittäisissä vaiheissa toimisi DIRKS:n pohjalta. Kuvioon on piirretty DIRKS:n esittämien yksittäisten iterointien lisäksi näkyviin myös säilytysaikojen määrittelyn ja asiakirjatyyppien kartoittamisen iteroinnit, jotka myös ovat mahdollisia.



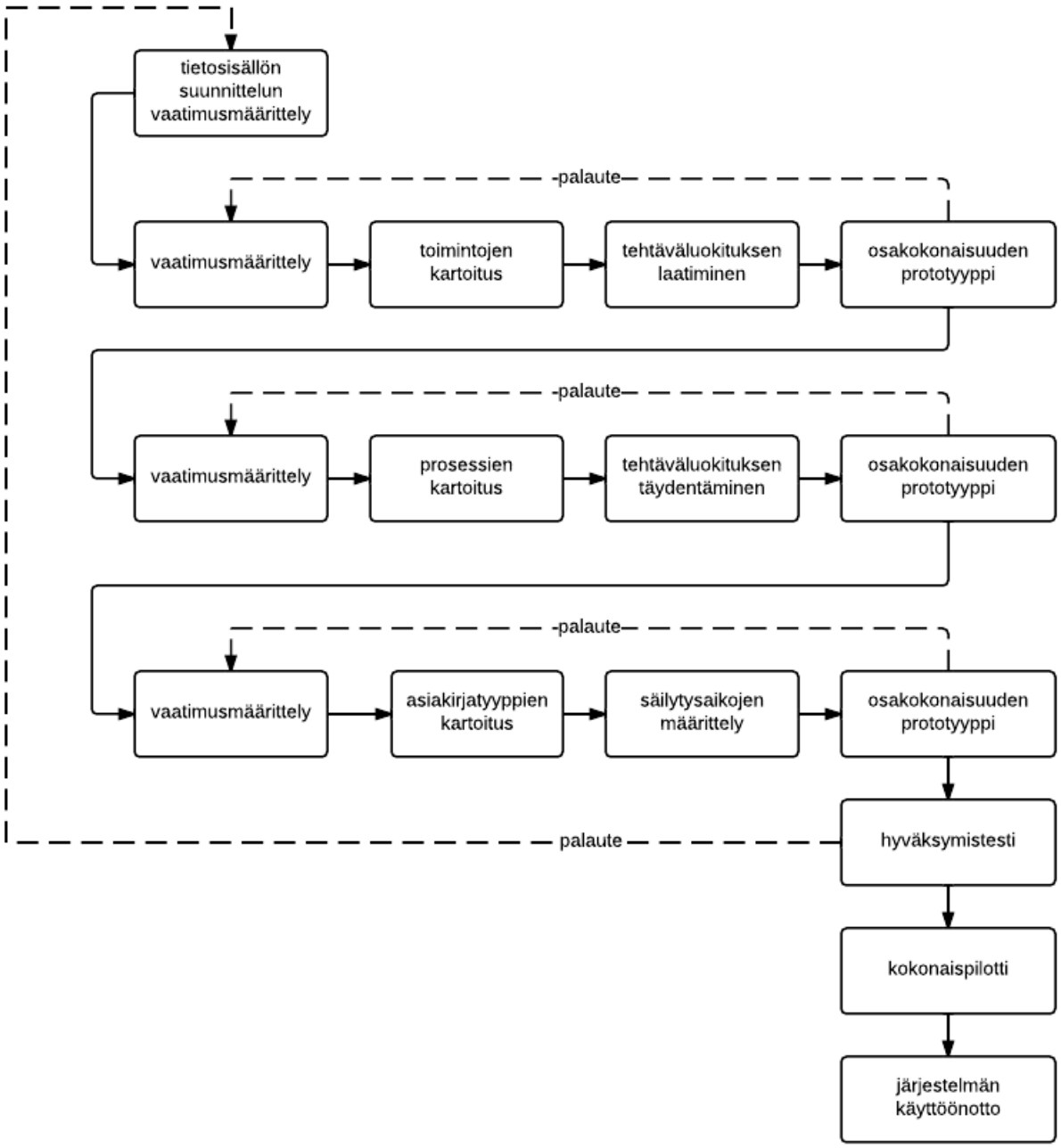
Kuvio 24. Suunnittelumalli yksittäisvaiheiden iteraatiolla

Ongelmana näissä lisäyksissä on se, että vaiheiden sisäiset palautesilmukat ja niiden läpikäynti voivat hidastaa koko järjestelmän tasolla saavutettavia välituloksia. Jos suunnittelussa jumitutaan pyörittämään yhden yksittäisen vaiheen iterointia niin kauan, että ollaan tyytyväisiä tulokseen, laajemman kokonaisuuden iterointi hidastuu. Toisaalta prototyyppien hyödyntäminen suunnittelussa tekee tällaisen iteroinnin luontevaksi. Vaiheiden testaamisessa prototyyp-

pien avulla ei ole mielekästä, jos tästä testaamisesta ei kerätä palautetta ja korjata järjestelmää palautteen perusteella iteratiivisesti.

Lisäksi Kautzin (2011, 227) tutkimuksen perusteella onnistuneessa suunnitteluprojektissa palautesilmukat on järkevää nimenomaan pyrkiä liittämään johonkin konreettiseen välituotokseen, kuten pilottiprojektiin tai prototyyppiin ja myös pienemmistä kokonaisuuksista voi tällä tavalla kerätä palautetta, jota hyödynnetään saman kokonaisuuden seuraavassa iteraatiossa. Pienempien kokonaisuuksien palautteen keräämisestä on Kautzin (2011, 228) mukaan etuna se, että käyttäjät saadaan näin aktiivisesti mukaan suunnitteluun ja palautetta kerättyä niin usein, että väärinkäsitykset ja suunnittelun virheet voidaan korjata nopeasti ennen kuin ne kasvavat suuremmiksi ongelmiksi.

Banslerin ja Havnin (2010, 638) mukaan iteraatioita sovelletaan informaatiojärjestelmien suunnittelussa usein myös niin, että peräkkäisten vaiheiden välillä on palautesilmukoita, jotka antavat tietoa siitä, miten aiempien vaiheiden toteutus on onnistunut ja tämän palautteen pohjalta voidaan palata aiempiin suunnittelun vaiheisiin ja korjata niitä. Bansler ja Havn tarkoittavat tosin näillä peräkkäisillä vaiheilla enemmän laajempaa suunnittelukokonaisuutta, jossa järjestelmän suunnittelu on vain yksittäinen vaihe, jota ei ole avattu oman mallini tavoin. Omassa mallissani on luontevampaa lisätä palautesilmukat osakokonaisuuksiin niin, että yksittäistä osakokonaisuutta koskevasta pilottiprojektista kerätään palautetta, joka vaikuttaa osakokonaisuuden vaatimusmäärittelyihin seuraavassa iteraatiossa. Lisäksi koko järjestelmän suunnittelun iteraatiolle lopussa toteutettavasta pilotista voidaan tehdä palautesilmukka uudella iteraatiolle koko suunnittelun vaatimusmäärittelyihin:



Kuvio 25. Palautesilmukat ja iterointi osakokonaisuuksien ja koko suunnittelun tasolla

Tässä kuviossa ei ole selkeyden vuoksi piirretty näkyviin kuviossa 24 näkyneitä yksittäisten vaiheiden palautesilmukoita ja prototyyppejä, mutta suunnittelun kaikille tasoille – yksittäisiin vaiheisiin, osakokonaisuuksiin ja koko tietosisällön suunnittelulle – on nyt mallissa lisätty jonkinlainen palautteenkeräysjärjestely (prototyyppi tai arviointi) ja tehty niistä palautesilmukat joko vaiheeseen itseensä tai vaatimusmäärittelyyn. Malli on näin melko monimutkainen. Koska tarkoituksena on kuitenkin kuvata sitä, miten suunnittelu konkreettisesti etenee, on perusteltua ottaa malliin mukaan myös aiemmassa kuviossa esitetyt yksittäiset eri suunnitteluvaiheiden palautesilmukat, koska ne ovat käytännön suunnittelun kannalta olennaisia. Ite-

roinnin lisääminen yksittäisiin vaiheisiin on perusteltua myös siksi, että nämä vaiheet, kuten prosessien kartoitus, ovat todellisuudessa varsin laajoja suunnittelukokonaisuuksia.

Iteraatiot on piirretty tarkoituksella nimenomaan vaatimusmäärittelyihin osakokonaisuuden ensimmäisen suoritusvaiheen sijaan. Ajatuksena on, että palautteen pohjalta vaatimusmäärittelyitä muokataan joustavasti suunnittelun edetessä. Evansin ja Rouchen (2006, 320–328) mukaan järjestelmän vaatimusten määrittely voidaan toteuttaa iteratiivisesti niin, että vaatimusmäärittelyä tarkennetaan ja kehitetään versio versiolta palautteen pohjalta. Joustavuuden lisääminen vaatimusmäärittelyjen tekemiseen on tärkeää, sillä Johnstonin (2005, 4) mukaan arviot ohjelmistoprojekteista osoittavat, että noin 85 % ohjelmistoissa olevista virheistä voidaan johtaa puutteellisiin vaatimusmäärittelyihin. Joustavuuden lisääminen vaatimusten määrittelemiseen edesauttaa osaltaan sitä, että järjestelmä vastaa paremmin niitä tarpeita, joita sen asiakkailla on, koska vaatimuksia voidaan tarkastaa paremmin ja muokata niitä projektin edetessä, kun saadaan lisäpalautetta käyttäjiltä ja ymmärretään vaatimukset paremmin.

Pilottiprojektien iteraatio on jätetty kuvio 25:stä tarkoituksella pois. Myös pilottiprojektista voi ja kannattaa sinällään tehdä iteraatio taaksepäin, mutta se on järkevää kohdistaa koko suunnittelun vaatimusmäärittelyyn, ei vain tietosisällön, koska pilottiprojektissa testataan järjestelmää kokonaisuudessaan, ei vain tietosisältöä. Koska järjestelmän laajempi suunnittelu ei kuulu tämän työn rajaukseen, jätin pilottiprojektin iteraation piirtämättä kuvion pitämiseksi yksinkertaisena.

Lisäksi on huomattava, että vaikka kuviossa iterointi vaikuttaa päättyvän järjestelmän käyttöönottoon, iterointia voidaan jatkaa myös järjestelmän käyttöönoton jälkeen. Iterointiin ja jatkuvaan kehittämiseen perustuva suunnittelufilosofia on hyvä lähtökohta sellaisten järjestelmien suunnitteluun, joita halutaan kehittää jatkuvasti sen sijaan, että ne vain kehitetään ja jätetään sen jälkeen asiakkaille sellaisenaan ilman jatkokehitystä. (Karlström & Runeson 2005, 44) Vaikka tässä työssä projektin loppupisteeksi on asetettu järjestelmän käyttöönotto päivittäiseen käyttöön, iteratiivinen kehittäminen voi jatkua myös aktiivisena käyttöönoton jälkeen ja auttaa järjestelmän jatkuvassa parantamisessa. Tämä pätee myös sähköisiin asiakirjajärjestelmiin, joiden yhteydessä yleensä korostetaan niiden jatkuvan kehittämisen tarvetta. Esimerkiksi AMS-oppaassa tuodaan esille asiakirjojen hallintavälineiden päivittämisen tarve, mutta lineaarisissa vesiputousmalliin perustuvissa projekteissa tällainen jatkuva kehittäminen on vaikeampaa kuin ketterän kehityksen menetelmiä hyödyntävissä projekteissa. Lineaarisia

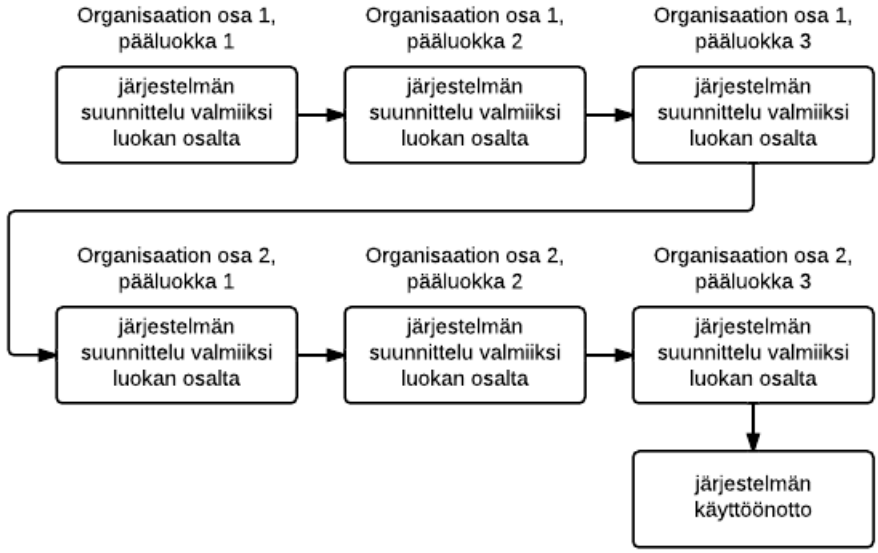
projekteja käsitellään yleensä yksittäisinä kokonaisuuksina, joille annetaan tietty määrä resursseja ja projektilla on yleensä selkeä lopetuspiste, jonka jälkeen kehittäminen vaatii usein kokonaan uuden projektin aloittamisen uusilla resursseilla.

Yksi erityinen ongelma aktiivisessa iteroinnissa on se, milloin järjestelmä on ”valmis”, eli milloin suunnittelun iterointi lopetetaan ja otetaan järjestelmä käyttöön. Tähän on lähestymistapoja ja tarpeiden mukaan voidaan päätyä hyvin erilaisiin ratkaisuihin. Keskeistä on se, että järjestelmän ensimmäisen käyttöversion valmistumiselle asetetaan selkeitä kriteereitä. Valmistumista voidaan rajata esimerkiksi ajan perusteella tai riittävän laadun saavuttamisen mukaan. Laatunäkökulmasta järjestelmän voidaan katsoa olevan valmis esimerkiksi siinä vaiheessa, kun käyttäjät ovat siihen riittävän tyytyväisiä. Toisaalta jos ketterän kehityksen periaatteita seurataan pidemmälle, voidaan lähteä siitä, että järjestelmää kehitetään jatkuvasti paremmaksi ja paremmaksi. Tällöin voidaan alussa hyväksyä joiltain osin puutteellinen järjestelmä ja parantaa sitä hiljalleen vielä käyttöönoton jälkeen.

4.3 Kokonaismalli

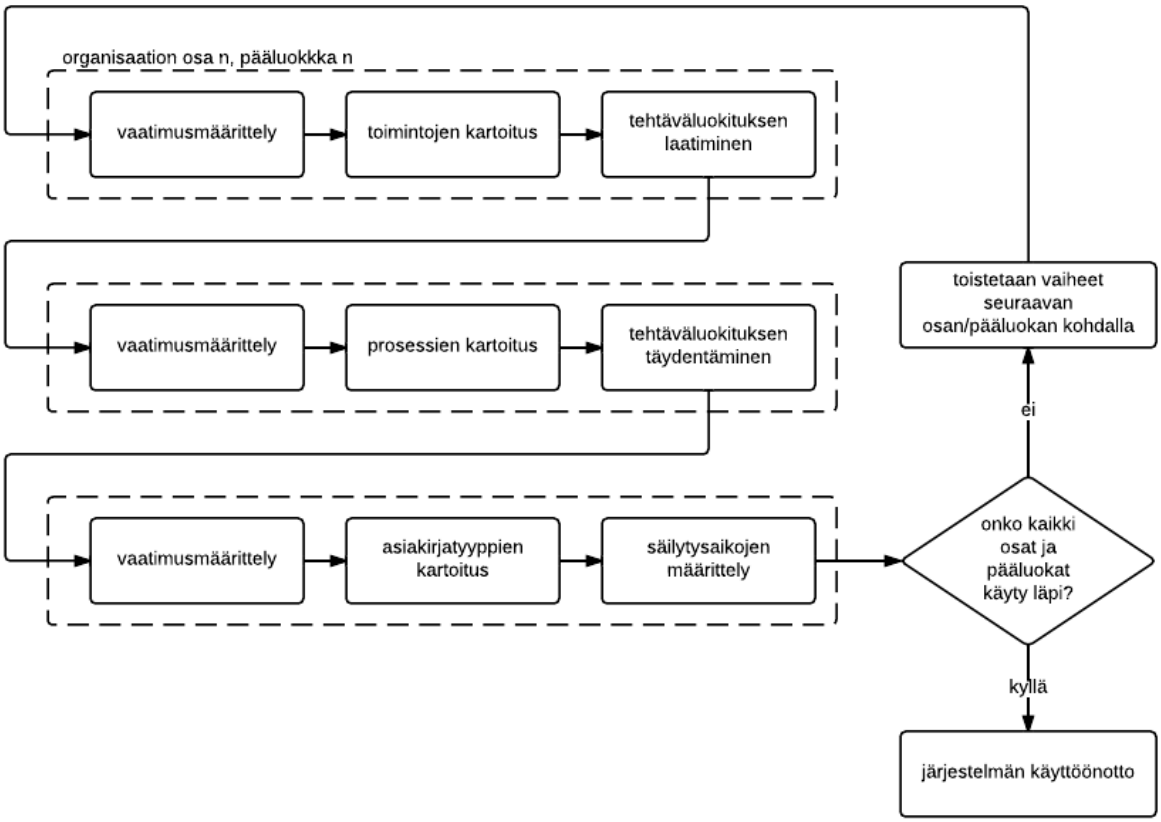
Edellisissä luvuissa olen esitellyt mallin kehittämistä yksittäisten ketterän kehityksen menetelmien pohjalta. Tässä luvussa sovitan aiemmin esittelemäni tavat soveltaa ketteriä kehitysmenetelmiä yhdeksi kokonaisuudeksi, jota kutsun kokonaismalliksi. Pyrin säilyttämään yksittäisiä ketterän kehityksen menetelmiä käsitelleissä luvuissa tehdyt ratkaisut sellaisenaan kokonaismallissa, jos se on mahdollista. Tarvittaessa pudotan kuitenkin aiemmin esiteltyjä sovellustapoja pois, jos ne eivät toimi järkevästi kokonaisuuden osana. Tarkoituksena kokonaismallin rakentamisessa on esitellä yksi mahdollinen tapa soveltaa ketteriä kehitysmenetelmiä koko sähköisen asiakirjajärjestelmän tietosisällön suunnittelussa. Kokonaismalli toimii lähinnä esimerkkinä menetelmien soveltamisesta ja auttaa hahmottamaan, miten työssä esitellyt yksittäiset ketterien kehityksen menetelmien sovellukset voisivat toimia kokonaisuutena laajemman suunnittelukokonaisuuden toteuttamisen pohjana. Kokonaismalli on rakennettu työssä painotettujen kriteerien, nopeuden ja joustavuuden, pohjalta ja pyrkii maksimoimaan nämä ominaisuudet suunnittelussa.

Otan kokonaismallin pohjaksi erilaiset projektin ositukset. Nopeuden maksimoimiseksi erityisesti suunnitteluprojektin alkuvaiheessa on perusteltua pilkkoa projekti pieniin kokonaisuuksiin. Otan lähtökohdaksi sekä organisaatioyksiköittäin jakamisen että jakamisen tehtäväluokituksen pohjalta pääluokittain, jonka esitin aiemmin osittamista käsittelevässä luvussa:



Kuvio 26. Esimerkki osittamisesta organisaation osiin ja pääluokkiin kahdella osalla ja kolmella pääluokalla

Kun suunnittelu ositetaan vielä organisaation osien ja pääluokkien lisäksi osakokonaisuuksiin, malli näyttää tältä:



Kuvio 27. Osittaminen organisaation osiin, pääluokkiin ja osakokonaisuuksiin

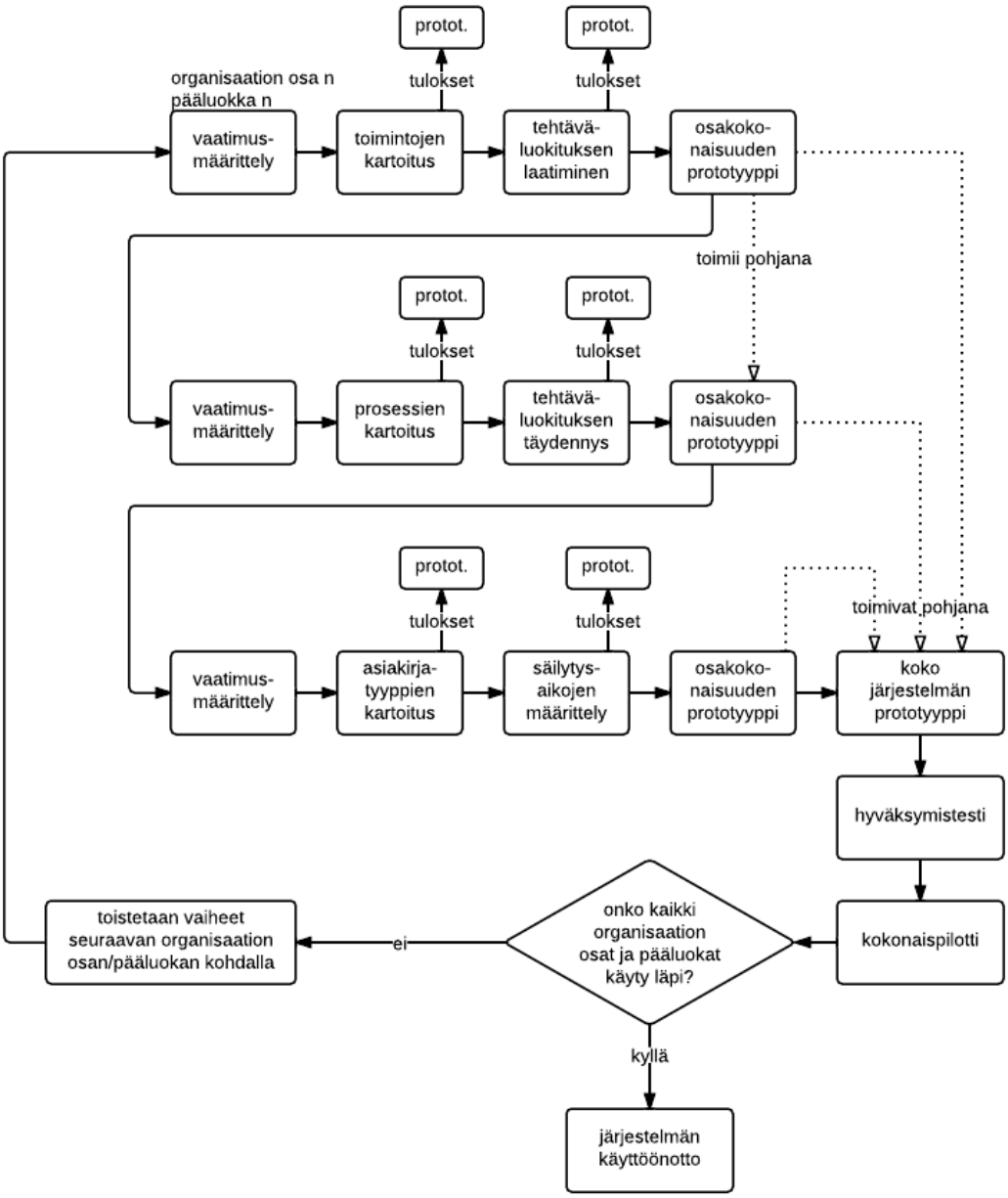
Mekaanisesti tätä mallia soveltamalla suunnittelu jakaantuu suurimmassa osassa organisaatioita todella pieniin kokonaisuuksiin. Käytännössä organisaation jakaminen osiin suunnittelua varten täytyy pitää yksinkertaisena ja pääluokittain jakamisessa esimerkiksi niputtaa pääluokkia yhteen ja hyödyntää pääluokittain osittamista vain silloin, kun halutaan pilkkoa suunnittelua pieniin osiin esimerkiksi kokemuksen hankkimiseksi tai suunnittelun kehittämiseksi.

Lisäksi kuten osittamisen käsittelyssä pohdin, pääluokittain osittamisessa on huomioitava, että se ei toimi tällaisessa iteratiivisessa ja joustavuutta ja nopeutta korostavassa mallissa lähtökohtaisesti kovin hyvin, koska tehtäväluokitus, jolle pääluokittain osittaminen pohjautuu, ei ole suunnittelun alkuvaiheessa vielä vakiintunut. Tämän takia pääluokittain osittaminen on usein järkevää rajata koko projektin tai yksittäisen organisaation osan tietosisällön suunnittelun alkuvaiheeseen, jolloin voidaan poimia muutamia niin keskeisiä tehtäväluokkia, että voidaan luottaa siihen, että ne pysyvät luokituksessa myös myöhemmillä suunnittelun iteraatiokierroksilla. Vaikka pääluokittain osittaminen ei yleensä ole järkevää koko suunnittelun laajuudessa, olen silti piirtänyt tehtäväluokittain osittamisen näkyviin kokonaismallin niin, että se koskee kaikkia vaiheita, koska periaatteessa tällainen osittaminen on mahdollista kaikissa suunnittelun vaiheissa, ja on tapauskohtaista miten laajasti ja missä paikoissa suunnittelua sitä voidaan hyödyntää mielekkäästi.

Seuraavaksi lisään malliin prototyypit, pilottiprojektit ja hyväksymisarvioinnin. Näiden käsittelyn yhteydessä päädyin siihen, että pienempien vaiheiden ja osakokonaisuuksien palautteen keräämiseen ja yksittäisten ideoiden testaamiseen prototyypit toimivat oletettavasti paremmin. Pilottiprojekteja voi periaatteessa toteuttaa myös osakokonaisuuksille, mutta niiden vaativuuden vuoksi tämä ei yleensä ole järkevää. Lisäksi hyväksymisarvioinnit ovat joustavampi ja nopeampi keino kokonaisen iteraation arvioimiseksi kuin pilotin toteutus, joten sovellan lopullisessa mallissa pilotointia vain viimeisessä kokonaisiteraatiossa koko järjestelmän testaamiseen ennen sen käyttöönottoa. Päädyin myös siihen, että prototyypit on pääosin järkevintä rakentaa toisistaan erillisinä paitsi osakokonaisuuksien prototyyppejä, joiden kohdalla tehtäväluokitukseen liittyvät osakokonaisuuksien prototyypit voi yhdistää ja samoin käyttää osakokonaisuuksien prototyyppejä koko järjestelmän käsittävän prototyypin laadinnassa.

Pilottiprojekteja voidaan kuitenkin toteuttaa useita, koska suunnittelua on ositettu organisaation osittain ja pääluokittain. Tämä ei välttämättä ole järkevää aiemmin esille tuodun pilottipro-

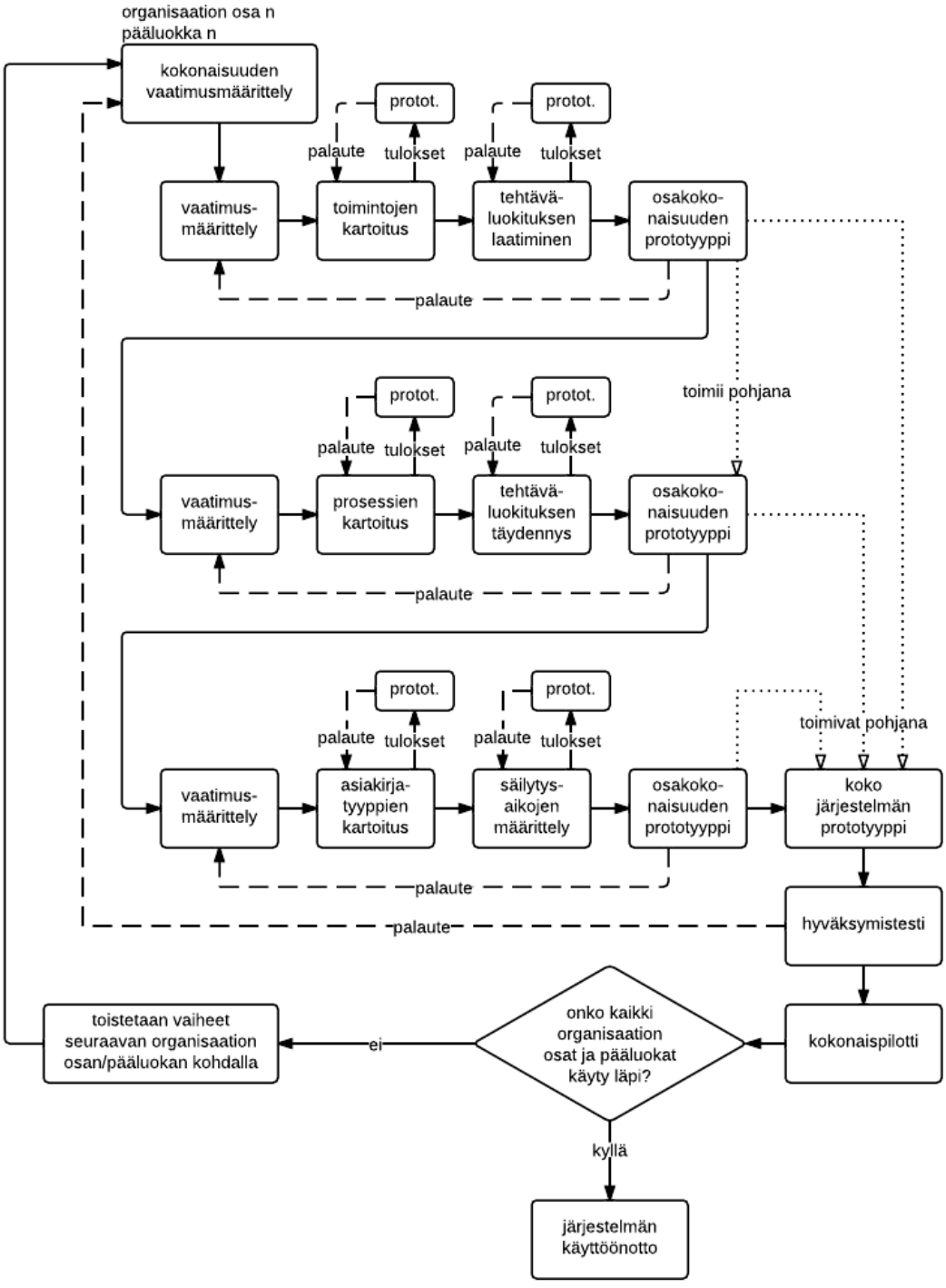
jektien raskauden vuoksi. Piirsin kuitenkin kuvioon pilottiprojektin jokaisen organisaation osan ja pääluokan kohdalle, koska pilottiprojekteja voidaan näissä kohti mielekkäästi soveltaa. Jos toteutetaan vain yksi pilottiprojekti koko projektin lopussa, kuviota pitäisi muokata niin, että seuraavaan osaan tai pääluokkaan siirryttäisiin suoraan hyväksymistestistä ja pilottiprojekti tulisi vasta kaikkien osien ja pääluokkien suunnittelun jälkeen ennen käyttöönottoa. Jättämällä mahdollisuuden pilottiprojektien toteuttamiseen myös yksittäisissä osien tai pääluokkien suunnittelun toteutuksissa, malli näyttää tältä:



Kuvio 28. Kokonaismalli pilottiprojekteilla ja prototyypeillä

Viimeisenä malliin lisätään vielä iteraatiot niin, että suunnittelu jakaantuu pienempiin ja ketterämpiin kokonaisuuksiin, joita voidaan iteroida ennen seuraaviin kokonaisuuksiin siirtymistä. Iteraatiot on mallissa sijoitettu niin, että ne suuntautuvat prototyyppien ja pilottiprojektien

laatimisesta yksittäisten osakokonaisuuksien vaatimusmäärittelyihin tai koko suunnittelun vaatimusmäärittelyyn. Lisäksi kuvioon on lisätty yksittäisten vaiheiden yhteyteen edellisestä kuviosta tarkoituksella pois jätetyt palautesilmukat takaisin prototyypeistä samaan vaiheeseen. Malliin on lisätty myös koko suunnittelukokonaisuuden vaatimusmäärittely, jossa asetetaan uuden iteraation tavoitteet ja vaatimukset, jotka pyritään toteuttamaan saadun palautteen pohjalta:



Kuvio 29. Kokonaismalli

Koska malli on ositettu niin voimakkaasti, näin kattava iterointi sekä prototyyppien ja pilotti-projektien hyödyntäminen käy nopeasti kohtuuttoman raskaaksi. Vaikka malli olisi suoraan sovellettuna erittäin joustava, mallin nopeus kärsii merkittävästi, jos toteutusta ei vaiheittain kevennetä. Käytännössä prototyyppien ja pilottiprojektien toteuttamista on järkevää vähentää koko ajan, kun edetään organisaation osasta toiseen ja pääluokasta toiseen. Iterointia ja prototyyppien laatimista voidaan vähentää esimerkiksi sitä mukaa kuin suunnitteluryhmän kokemus ja ymmärrys projektin toteuttamisesta lisääntyy. Iterointia ja prototyyppien tuottamista ei kuitenkaan joka tapauksessa kannata missään vaiheessa jättää kokonaan tekemättä, koska eri organisaation osat ja niiden vaatimukset ovat erilaisia.

5 MALLIN VAIKUTUS SUUNNITTELUN HALLITTAVUUTEEN

Tässä luvussa pohdin kirjallisuuden pohjalta sitä, miten kehitetty malli vaikuttaa kokemattoman projektiryhmän mahdollisuuksiin hallita sähköisen asiakirjajärjestelmän tietosisällön suunnittelutyötä. Käsittelen hallittavuutta kahdessa osassa tutkimusasetelmassa esittämäni hallittavuuden määritelmän pohjalta. Ensimmäisessä alaluvussa pohdin sitä, miten malli vaikuttaa suunnittelijoiden kykyyn oppia suunnittelun aikana paitsi yleisesti suunnittelusta ja sen toteuttamisesta myös suunnitteluprojektin vaatimuksista ja käyttäjien tarpeista. Oppimiseen liittyen tarkastelen myös sitä, miten malli mahdollistaa erilaisten suunnittelukokeilujen tekemisen verrattuna lineaarisiin malleihin. Toisessa alaluvussa käyn läpi sitä, miten malliin lisätty joustavuus ja sen kriteerinä toiminut käyttäjälähtöisyys vaikuttavat projektin hallittavuuteen.

5.1 Oppiminen, kokeilujen tekeminen ja uudet osaamisvaatimukset

Kirjallisuudessa puhutaan usein niin sanotusta virheiden tai vikojen taloudesta, kun käsitellään sitä, kuinka kalliiksi suunnittelijoiden tekemät virheet tulevat suunnittelun eri vaiheissa. Ajatus on, että virheen hinta kasvaa sitä suuremmaksi, mitä myöhäisemmässä vaiheessa projektia se havaitaan ja kuinka vaikeaa sitä on korjata. Esimerkiksi lineaarisessa vesiputousmallissa virheiden hinta kasvaa helposti korkeaksi, koska virheet havaitaan usein vasta suunnittelu loppuvaiheessa tai pahimmillaan vasta käyttöönoton jälkeen. Virheitä on lineaarisessa suunnittelussa vaikeaa havaita, koska palautteen kerääminen on puutteellista ja eri vaiheiden toteutusta ei testata ennen projektin loppua. Lisäksi vesiputousmallissa oletetaan, että vaatimusmäärittelyitä ei projektin aikana muokata, vaikka todellisuudessa vaatimukset muuttuvat lähes kaikissa projekteissa, kun suunnittelijat oppivat lisää järjestelmästä ja käyttäjien tarpeista. Tämän seurauksena vesiputousmallissa käy usein niin, että kalliiden virheiden lisäksi lopullinen tuote ei vastaa asiakkaan tarpeita, koska vaatimuksia ei tarkenneta projektin edetessä. (Johnston 2005, 2-3)

Kokemattoman projektiryhmän kannalta on erityisen tärkeää, että virheen hinta saadaan pidettyä matalana, koska virheiden riski on jo lähtökohtaisesti korkea. Mallissa käytetyistä ketterän

kehityksen menetelmistä iterointi ja järjestelmän testaaminen pilottiprojekteilla ja prototyypeillä sekä käyttäjien arviointi auttavat havaitsemaan virheet aikaisin ja laskevat niiden hintaa. Toinen hyöty erityisesti iteroinnista yhdessä arvioinnin kanssa on se, että ne mahdollistavat vaatimusmäärittelyjen tarkentamisen iteraatioiden välillä. Kokemattomienkaan suunnittelijoiden tekemät virheet ja puutteet vaatimusmäärittelyissä eivät tule niin kalliiksi, kun suunnittelu vaiheistetaan ja muutetaan iteratiiviseksi. Tällöin virheet kohdistuvat paitsi pienempiin kokonaisuuksiin, ne on myös mahdollista korjata nopeammin.

Myös suunnittelun osittaminen voimakkaasti edistää virheiden hinnan pienentämistä. Pienempiin osiin jaettuna yksittäiset virheet vaikuttavat rajallisemmin kokonaisuuteen ja ne voidaan korjata nopeammin. Esimerkiksi yhtä organisaation osaa tai pääluokkaa koskevat iteraatiot ovat huomattavasti suppeampia kuin koko organisaation kattavat. Osittaminen lisää myös suunnittelun hallittavuutta, koska se jakaa yksittäiset kokonaisuudet pieniin osiin ja suunnittelijat voivat keskittyä kerrallaan hallittavampaan kokonaisuuteen verrattuna siihen, että he suunnittelisivat kokonaisuutta yhdellä kertaa.

Voimakas osittaminen lisää kuitenkin helposti suunnitteluprojektin monimutkaisuutta, koska työ jakaantuu yhden selkeästi etenevän kokonaisuuden sijaan moniin pienempiin kokonaisuuksiin ja varsinaisten suunnitteluvaiheiden toteuttamisen ohella on kyettävä hallitsemaan myös eri osien muodostamaa kokonaisuutta. Suunnittelumallin soveltaminen on pilkotussa mallissa haastavampaa myös siksi, että suunnittelijat joutuvat tekemään päätöksiä siitä, miten suunnittelua kevennetään osista toiseen siirryttäessä. Mallin soveltaminen sellaisenaan ei ole kaikissa organisaation osissa järkevää, koska projekti jakaantuisi tällä tavalla todella pieniin palasiin. Lisäksi sovellettavien menetelmien määrää on hiljalleen vähennettävä projektin edistyessä, että kokonaisuus ei paisu kohtuuttoman monimutkaiseksi ja hidastu liikaa. Tämä kaikki vaatii suunnittelijoilta päätöksiä mallin soveltamisesta suunnittelun eri vaiheissa. Verrattuna lineaarisiin yhdessä vaiheessa kiinteällä kaavalla suoritettaviin malleihin, ketteriin kehitysmenetelmiin nojaava ositettu malli vaatii enemmän suunnittelun muokkaamista tarpeiden ja muuttuvien vaatimusten mukaan, mikä asettaa haasteita projektin hallittavuudelle. Lisäksi laajemminkin ketterään kehittämiseen perustuvassa suunnittelussa erilaisia arvioitavia vaihtoehtoja on paljon. Erityisesti pilottiprojektit, prototyypit ja iteratiivisuus lisäävät helposti liikaa käytettyinä projektiin käytettyä aikaa ja suunnitteluryhmän on kyettävä punnitsemaan menetelmien käyttöä niin, että liian pieniin osiin jakautunut työ yhdessä ketterän kehityksen menetelmien laajan soveltamisen kanssa ei käy liian raskaaksi.

Osittaminen kuitenkin edistää merkittävästi suunnittelijoiden oppimista, koska se jakaa suunnittelun pienempiin osiin ja mahdollistaa esimerkiksi yhdessä organisaation osassa tehtäväluokituksen laatimisesta saatujen kokemusten hyödyntämisen seuraavassa osassa. Suunnittelijat pystyvät pienempiin kokonaisuuksiin ositetussa projektissa keräämään kokemuksia aiemmista vaiheista ja reflektoimaan omaa suunnitteluaan näiden kokemusten pohjalta ja kehittämään suunnitteluteknikkaansa. Osittamattomissa suunnittelumalleissa tällainen reflektoiminen voi tapahtua pääosin vasta koko projektin valmistuttua ja askel askeleelta tapahtuva oppiminen ei ole samalla tavalla mahdollista kuin ositetussa mallissa.

Prototyyppien ja pilottiprojektien toteuttaminen antaa suunnittelijoille mahdollisuuden paitsi oppia käyttäjien tarpeista, myös peilata omia ajatuksiaan ja suunnittelutapoja todellisiin käyttäjien asettamiin vaatimuksiin ja arvioida, miten hyvin heidän suunnitelmansa vastaavat todellisia tarpeita. Rzevskin (1984) mukaan lisääntynyt yhteys käyttäjiin on projektin hallinnan oppimisen ohella pilottiprojektien ja prototyyppien tärkein opettava tekijä. Käyttäjiltä saadun palautteen pohjalta suunnittelijat voivat peilata omia näkemyksiään järjestelmän suunnittelusta ja oppia suunnittelemaan käyttäjälähtöisiä järjestelmiä.

Pilottiprojektien osalta Glass (1997, 86) on todennut, että niiden toteuttamista pitäisi ajatella jo lähtökohtaisesti tutkimus- ja kehittämisprojekteina. Tutkimuksellisuus auttaa havaitsemaan ongelmia suunnittelussa ja testaamaan erilaisia suunnitteluvaihtoehtoja saadun palautteen pohjalta ja toimii hyvänä oppimisvälineenä suunnittelijoille. Pilottien erityinen etu on se, että niissä voidaan tarkastella tehtyjen suunnitteluratkaisujen toimivuutta aitoa lähelle vastaavassa ympäristössä. Oppimisen kannalta pilottiprojektien toteutus voi tiedonkeräämisen lisäksi olla hyödyllistä, koska se antaa paitsi monipuolisempaa osaamista projektienhallinnasta myös valmistaa suunnittelijoita valmiin järjestelmän käyttöönottoon ja siihen liittyviin haasteisiin (Bansler & Havn 2010, 640).

Glassin (1997, 88–89) mukaan huonosti suunnitellut minipilottiprojektit ovat kuitenkin yleensä hyödyttömiä, koska niistä ei opita paljoa. Myös Bansler & Havn (2010 638–640) ovat huomauttaneet, että pilottiprojekteista ei ole yleensä suurta hyötyä, jos niiden toteutus tehdään huolimattomasti. Usein oletetaan, että pilottiprojektit ovat yksinkertaisempia ja riskittömämpiä kuin tavalliset projektit, mutta tämä ei usein pidä paikkaansa. Paitsi että pilottiprojektit

ovat haastavia itsessään toteuttaa, ne vaativat erilaista osaamista kuin tavalliset projektit. (Bansler & Havn 2010, 637–640)

Riski pilottiprojektien epäonnistumisesta on ongelmallista myös siksi, että virheiden hinta on pilottiprojekteissa melko korkea. Pilottiprojektit vaativat huomattavasti resursseja, koska jokainen pilottiprojekti on itsessään laaja suunnitteluprojekti. Pilottiprojektien suunnittelun haastavuutta voi vähentää soveltamalla siihenkin ketteriä kehitysmenetelmiä, koska pilottiprojektien toteuttaminen vastaa monilta osin laajemman suunnitteluprojektin toteutusta. Tosin, kuten Bansler ja Havn totesivat, pilottiprojektit vaativat monelta osin myös erilaista osaamista ja näin esimerkiksi tässä työssä esitellyn mallin soveltaminen suoraan pilottien toteuttamiseen ei toimi. Pilottiprojektin toteuttaminen ketteriä kehitysmenetelmiä toteuttamalla voi myös usein olla liian kallista ja siksi epärealistista, vaikka se voisikin olla kokemattomille suunnittelijoille hyvä lähestymistapa.

Vaikka pilottiprojektit ovat periaatteessa hyvä tilaisuus oppia projektien toteuttamisesta aidossa ympäristössä ja kerätä järjestelmällisesti palautetta, pilottiprojektien toteuttamisen haastavuus on ongelmallista kokemattomille suunnittelijoille, joiden pitäisi pystyä toteuttamaan pilottiprojekti riittävän onnistuneesti, että siitä pystyttäisiin oppimaan jotain. Pilottiprojektit voisivat toimia tämän työn mallia paremmin oppimisen välineinä, jos niitä toteutettaisiin suunnittelussa useammin, jolloin suunnittelijoiden osaaminen niiden toteuttamisesta lisääntyisi ja piloteista saataisiin enemmän irti. Esimerkiksi minipilottien avulla voitaisiin saada kokemusta laajemman pilotin toteuttamisesta, mutta tässä mallissa niitä ei käytetä. Toisaalta yksinäänkin toteutettuna tietosisällön suunnittelun lopussa toteutettu pilottiprojekti voi onnistuessaan antaa suunnittelijoille kokemusta varsinaisesta koko järjestelmän pilotoinnista ja järjestelmän käyttöönotosta.

Jos pilottiprojektien hyöty oppimisen kannalta on tässä mallissa rajallinen, prototyyppien aktiivinen hyödyntäminen yhdessä iteroinnin kanssa ovat mielestäni oppimisen kannalta tämän mallin vahvuus verrattuna lineaarisiin malleihin, joissa käyttäjäpalautteen kerääminen tapahtuu suunnittelijalähtöisesti erilaisilla esittelyillä ja kyselyillä, koska suunnittelun tukena ei ole konkreettisia välineitä, joilla järjestelmän toimintaa voitaisiin esitellä. Palautteen keräämisen ohella prototyypit toimivat tehokkaana keinona kokeilla erilaisia lähestymistapoja ongelmiin. Prototyyppien hyödyt oppimisessa liittyvät pääosin jo edellisessä luvussa käsiteltyyn kokeilujen mahdollistamiseen ja virheiden havaitsemiseen mahdollisimman aikaisin.

Prototyypeillä tapahtuva joustava uusien ideoiden testaaminen ja kilpailevien suunnitteluratkaisujen vertaaminen laskee huomattavasti virheiden hintaa verrattuna siihen, että näitä erilaisia lähestymistapoja jouduttaisiin kokeilemaan esimerkiksi eri projekteissa, kun on ensin huomattu, että aiemmin käytetty suunnitteluratkaisu ei toimi. Usein pelätään vaihtoehtoisten ratkaisujen testaamiseen menevien resurssien paljoutta, mutta Collyer et al. (2010, 117) huomauttavat, että epäonnistuneet kokeilut opettavat tekijöitään usein yhtä paljon kuin onnistuneet ja kilpailu erilaisten lähestymistapojen välillä on tehokas keino lisätä ideoiden tuottamista. Heidän mukaansa kustannusten vastapainona ideoiden testaaminen parantaa laatua ja tuottaa kustannussäästöjä, kun kaikkein tehokkaimmat toimintatavat löydetään mahdollisimman varhain.

Lineaarisilla menetelmillä toteutetuissa projekteissa kynnys epävarmojen suunnitteluratkaisujen tekemiseen nousee, koska mahdollisten virheiden hinta voi nousta korkeaksi. Paineet ovat kovat, koska järjestelmä täytyy saada valmiiksi tarkkaan määriteltyjen ennakkosuunnitelmien mukaan ja järjestelmän käyttöönoton jälkeen korjausvaraa ei ole paljoa. Jos kokeiluihin suhtaudutaan myönteisemmin ja pidetään epäonnistuneita kokeiluja hyvinä, koska niiden avulla vältettiin huono lähestymistapa ja yhdistetään suunnitteluun ketteriä menetelmiä, paineet vähenevät ja suunnittelijat voivat luottaa siihen, että he voivat aina parantaa mallia ja kehittää sitä tarpeen mukaan joustavasti suunnittelun edetessä vielä valmistumisen jälkeenkin.

Prototyyppien tavoin myös suunnittelun arviointi jokaisen iteraation lopussa lisää saadun palautteen ja reflektoinnin kautta suunnittelijoiden oppimista. Remenyin ja Sherwood-Smithin (1999, 19) mukaan suunnittelun arviointi edistää suunnittelijoiden oppimista järjestelmän vaatimusten toteutumisesta. He eivät tällä ”oppimisella” varsinaisesti tarkoita suunnittelijoiden osaamisen yleisempää lisääntymistä, vaan tiedon saamista vaatimusten täyttymisestä. Silti malliin lisätty testaaminen jokaisen kokonaisiteraation lopussa edistää myös laajemmin oppimista suunnittelun toteuttamisesta, sillä se tarjoaa keinon peilata jokaisen iteraation suunnittelun onnistumista suhteessa asetettuihin vaatimuksiin. Tällä tavalla suunnittelijat voivat arvioida omaa toimintaansa ja sen onnistumista suhteessa todellisiin tarpeisiin ja kehittää suunnittelumenetelmiään palautteen pohjalta.

Mallissa sovelletuista ketterän kehityksen menetelmistä myös iterointi mahdollistaa yhdessä prototyyppien kanssa joustavan ratkaisumallien testaamisen. Esimerkiksi tehtäväluokituksen

osalta prototyypit ja iterointi mahdollistavat erilaisten luokitusmallien testaamisen, mikä lineaarisessa suunnittelussa olisi vaikeaa, koska vaatimusmäärittelyt lyödään yleensä lukkoon jo projektin alkuvaiheessa. Tehtäväluokituksen laatimisen yhteydessä voisi esimerkiksi kokeilla syvempien tai matalampien luokitusten toimivuutta organisaatiossa ja käyttäjien suhtautumista erilaisiin ratkaisuihin.

Iteroinnin etuna on erilaisten lähestymistapojen testaamisen ohella se, että se antaa mahdollisuuden tulosten reflektointiin. Peter Senge (1994, 303) on esittänyt, että organisaatioissa tapahtuvassa kehittämistyössä jatkuva hypoteesien kehittäminen, niiden testaaminen käytännössä ja tulosten reflektointi ovat olennaisia uuden tietämyksen kehittämisen kannalta. Lineaarisesti etenevät ongelmanratkaisumallit eivät jätä tilaa rohkealle kokeilulle ja erilaisten lähestymistapojen testaamiselle. Testaamalla erilaisia toteutustapoja ja reflektoimalla niiden tuloksia, voidaan oppia lisää siitä, miten erilaisia ongelmia voitaisiin lähestyä. Iterointi laskee kynnystä tehdä kokeiluja. Lineaarisessa vesiputousmallin mukaisessa kehittämisessä asiat on sen sijaan saatava kerralla oikein. Paitsi että ajatus on epärealistinen, se estää erilaisten lähestymistapojen testaamisen ja kehittämisen, ja rohkaisee tekemään asioita niin kuin niitä on tehty aina ennenkin.

Ketterän kehityksen menetelmien ja erityisesti iteroinnin käyttäminen edistävät suunnittelun hallittavuutta myös sitä kautta, että ne suhtautuvat kehittämiseen hyvin eri tavalla kuin lineaariset mallit. Ketterässä kehittämisessä ajatellaan yleensä jo lähtökohtaisesti, että järjestelmä ei välttämättä valmistu ollenkaan kehityksen menetelmien avulla. Ajatuksena on, että kehittämistä jatketaan koko järjestelmän elinkaaren ajan, ei vain laajoina silloin tällöin käynnistettävinä erillisinä projekteina, joissa järjestelmä uusitaan tai sitä päivitetään. Vaikka työssäni rajasin tarkastelun päättymään järjestelmän käyttöönottoon, tämä ei tarkoita, ettei kehittäminen tämän jälkeen jatkuisi.

Iteratiivisen suunnittelumallin etuna on se, että se rakentuu jo lähtökohtaisesti jatkuvan kehittämisen varaan ja jakaa suunnittelua osiin. Kun järjestelmää kehitetään jo käyttöönottoa edeltävässä suunnittelussa toistensa päälle rakentuvina iteraatioina, kynnys käytössä olevan järjestelmän ja asiakirjahallinnan välineiden jatkuvaan kehittämiseen laskee. Työssä esitelty malli soveltuu joustavasti järjestelmän kehittämiseen valmistumisen jälkeen ilman, että joudutaan käynnistämään uutta suunnitteluprojektia aina kun järjestelmää päivitetään. Järjestelmän kehittämistä vain jatketaan käyttöönoton jälkeen asettamalla uusia suunnittelutavoitteita ja suo-

rittamalla uusia iteraatioita. Iteraatioista voidaan lisäksi tehdä laajempaan suunnitteluprojektiin verrattuna niin hallittavia ja ketteriä, että kehittämisen yhden syklin pyöräytys olisi riittävän matalan kynnyksen takana ja helposti toteutettavissa ilman erillistä raskasta järjestelmän päivitysprojektia.

Myös lineaarisiin malleihin pohjautuvissa ohjeistuksissa korostetaan yleensä koko ajan joustavuutta ja tarvetta päivittää aktiivisesti erilaisia asiakirjahallinnan ohjausvälineitä. On kuitenkin epätodennäköistä, että asiakirjahallinnan välineitä pystytään ainakaan hyvin dynaamisissa ympäristöissä päivittämään riittävää tahtia, jos koko suunnitteluprosessia ei laadita niin, että se tukee tällaista jatkuvaa kehittymistä. Päivittämisen lisäksi suunnittelijoiden pitäisi pystyä vastaamaan nopeasti muuttuviin tarpeisiin ja kehittämään järjestelmää vaihe vaiheelta koko ajan sitä mukaa, kun tarpeet organisaatioissa muuttuvat.

Suunnittelijoiden oppimisen kannalta iteratiivisuus on suunnittelun rakenteena hyvä lähtökohta, koska siinä aiemmista iteraatioista saatuja kokemuksia voidaan hyödyntää ja lähestymistapoja kehittää seuraaviin iteraatioihin. Bansler ja Havn (2010, 638) ovat todenneet, että informaatiojärjestelmien suunnittelu on oppimisprosessi, jossa iteratiivisella lähestymistavalla opitaan koko ajan enemmän järjestelmän toteuttamisesta. Iteratiivisuuden päällekkäisyys ikään kuin jakaa suunnittelun toistensa päälle rakentuviksi kerroksiksi, joista jokaisessa suunnittelua voidaan kehittää saadun palautteen pohjalta sen sijaan, että jouduttaisiin odottamaan koko projektin valmistumista ennen palautteen saamista. Iteraatiot ovat oppimisen kannalta tavallaan kuin projekteja projektin sisällä ja mahdollistavat kehittymisen taso tasolta projektin edetessä.

Iteratiivisten ja muiden ketterien kehityksen menetelmien hyötynä voi olla suunnittelijoiden oppimisen kannalta myös se, että niiden avulla on helpompaa antaa suunnittelijoille vastuuta laajemmista kokonaisuuksista. Collyerin (2011, 118) mukaan ketterän kehityksen menetelmät tukevat hyvin suunnitteluprosessin hallintaa niin, että suunnittelulle asetetaan vain löyhät kehyssuunnitelmat ja keskitytään asettamaan tarkastuspisteitä ja laajempia tavoitteita tiukan ohjaamisen ja suoritettavien työvaiheiden määrittelyn sijaan. Tällainen liikkumavara vapauttaa suunnittelijat pohtimaan laajemmin parhaita lähestymistapoja tavoitteen saavuttamiseksi. Vastuunantaminen edistää suunnittelijoiden oppimista, koska he joutuvat miettimään tavoitteiden saavuttamista laajemmassa mittakaavassa sen sijaan, että toteuttaisivat vain hyvin rajattuja kokonaisuuksia tarkasti määriteltyjen ohjeiden pohjalta.

Ongelmana suunnittelijoiden vapauden korostamisessa ja tavoitteiden määrittelyssä tarkan ohjaamiseen sijaan saattaa kuitenkin olla se, että Collyerin et al. (2010, 119) mielestä tällainen väljempi suhtautuminen suunnittelun ohjaamiseen lisää kyllä joustavuutta, mutta vähentää ennakoivuutta. Tämän ennakoitavuuden uhraaminen puolestaan lisää heidän mukaansa usein tarvetta kokeneempien työntekijöiden käyttämiseen, että projekti pysyy hallinnassa. Voi olla, että hyvin väljästi asetetut tavoitteet voivat olla ongelmallisia kokemattomille suunnittelijoille. Näin lienee ainakin silloin, jos suunnittelijoilla ei ole aikaa oppia projektin edetessä suunnittelun vaatimuksista.

5.2 Joustavuuden ja käyttäjänäkökulman vaikutus hallittavuuteen

Asiakaslähtöisyyden korostaminen mallissa vaikuttaa myös hallittavuuteen. Yleisesti ketterien kehitysmenetelmien on todettu lisäävän suunnitteluryhmien kommunikaatiota (Karlström & Runeson 2005, 46). Lisääntynyt tiedonvaihto edistää suunnittelijoiden oppimista ja kehittymistä suunnittelun edetessä. Lisäksi Kautzin (2011, 217–218) tutkimuksen mukaan ketterien kehitysmenetelmien hyödyntäminen lisää kommunikaatiota suunnittelijoiden ja käyttäjien välillä. Myös Karlström ja Runeson (2005, 47) havaitsivat tutkimuksessaan, että ketterät kehitysmenetelmät ja erityisesti iterointi lisäävät lähes aina käyttäjiltä saatavan palautteen saamista. Tämä edistää heidän mukaansa myös suunnittelijoiden oppimista ja ongelmien sisäistämistä suunnittelun aikana.

Ketteriin kehitysmenetelmiin liittyy yleensä käyttäjien merkityksen korostaminen suunnittelussa. Tämä edesauttaa heidän tuomistaan konkreettisemmin mukaan kehittämistyöhön. Monet ketterästä kehityksestä kirjoittaneet pitävät nimenomaan käyttäjä- tai asiakaslähtöisyyttä keskeisimpänä ketterää kehitystä kuvaavana tekijänä. Ketterässä kehityksessä tavoitteena on tuoda käyttäjät aidosti mukaan suunnitteluun ja antaa heille todellisia mahdollisuuksia vaikuttaa suunnitteluun. Tästä on palautteen saamisen lisäksi se hyöty, että parhaimmillaan suunnittelijat saavat projektiinsa osaamista ja kokemusta suunnitteluun osallistuvien käyttäjien kautta. (Kautz 2011, 218)

Kokeneempien käyttäjien antama tuki ja ideat suunnittelutyössä ovat erityisen arvokkaita kokemattomien suunnittelijoiden oppimisen kannalta. Käyttäjät auttavat suunnittelua paitsi antamalla tietoa myös usein konkreettisesti opastamalla suunnittelijoita ja jakamalla heille osaamistaan. Esimerkiksi Kautzin (2011, 225, 230) tutkimassa ohjelmistoprojektissa kävi niin, että suunnitteluun osallistunut käyttäjä, jonka osaaminen tuki erityisen hyvin suunnittelutyötä, siirtyi projektin aikana täyspäiväisesti tukemaan projektin toteuttamista suunnittelijana. Tiiviisti suunnitteluun sidotut käyttäjät voivat Kautzin (2011, 217) mukaan toimia eräänlaisina apusuunnittelijoina. Käyttäjät vaikuttavat jatkuvan palautteen kautta aktiivisesti suunnitteluun ja voivat käyttäjäroolinsa ohella jakaa suunnittelijoille tietoa suunnittelun eri osa-alueista ja toimia konsultteina suunnittelutehtävissä.

Tähän liittyen Snyder ja Cox (1985, 67) ovat puolestaan todenneet, että tiivis yhteys käyttäjiin mahdollistaa suunnitteluun vaikuttavien toimintaympäristön muutosten tunnistamisen. Tunnistamalla erilaiset mahdolliset muutokset voidaan järjestelmän suunnittelua ohjata niin, että muutoksiin voidaan varautua ja muuttaa järjestelmän suunnittelua joustavasti myös järjestelmän rakentamisen aikana. Erilaiset palautesilmukat suunnittelussa mahdollistavat tällaisten muutostekijöiden tunnistamisen käyttäjien avulla ja toisaalta käyttäjien informoimisen suunnittelun muutoksista. (emt.) Suunnittelun hallittavuuden kannalta tämä muutosten tunnistaminen on hyödyllistä, koska se varmistaa, että muuttuvat tarpeet voidaan tunnistaa ja muuttaa suunnittelua niiden asettamien vaatimusten mukaiseksi. Lineaarisessa suunnittelussa on suurempi riski, että järjestelmää suunnitellaan loppuun asti alkuperäisten vaatimusmäärittelyiden mukaan, vaikka tarpeet muuttuvat.

Tieto myös liikkuu suunnittelussa kahteen suuntaan. Suunnittelijoiden paineita voi vähentää merkittävästi myös se, että tiiviiden yhteyksien ansiosta, myös käyttäjien käsitys projektista ja sen tavoitteista on realistisempi, koska he saavat osallistumisensa ja jatkuvan kommunikaation kautta paremmin tietoa projektin edistymisestä. Samalla mahdollisuus osallistua suunnitteluun ja vaikuttaa tehtyihin suunnittelupäätöksiin, vähentää muutosvastarintaa. (Kautz 2011, 220) Projektin hallittavuuden kannalta on hyödyllistä, että käyttäjillä on koko ajan realistinen kuva suunnittelusta ja sen tavoitteista. Downing (2006, 46) on todennut, että on mahdotonta kommunikoida liikaa käyttäjien kanssa siitä, mitä sähköisen asiakirjajärjestelmän käyttöönotossa tapahtuu. Tämä pätee epäilemättä myös järjestelmän suunnitteluun. On tärkeää kuvata järjestelmää mahdollisimman realistisesti, eli kertoa avoimesti paitsi toteutettavista myös poisjätettävistä ominaisuuksista.

Tässä työssä esitellyssä mallissa käyttäjät pyritään pitämään ajan tasalla paitsi erilaisten palautesilmukoiden ja niihin liittyvän keskustelun avulla, myös osallistamalla heidät aktiivisesti järjestelmän suunnitteluun ja testaamiseen suunnittelun kaikissa vaiheissa. Tämä on mielestäni parempi tapa tuoda käyttäjät mukaan projektiin ja pitää heidät ajan tasalla järjestelmän toteutuksesta kuin Downingin korostama aktiivinen tiedottaminen käyttäjille. Pelkkä tiedottaminen on yksipuolista ja lähtee suunnittelijoiden oletuksista siitä, mitä käyttäjien tulee tietää.

Lisäksi mikään ei varmista, että käyttäjät todella ymmärtävät tiedottamisesta huolimatta millainen järjestelmästä todella on tarkoitus tehdä. Tämän seurauksena he eivät osaa antaa palautetta järjestelmän kehittämisestä, mikä heikentää suunnittelijoiden kykyä oppia käyttäjiltä. Esimerkiksi Banslerin ja Havnin (2010, 637–638) mukaan käyttäjien palautteen kerääminen ei ole yksinkertaista, koska käyttäjät eivät ymmärrä abstrakteja järjestelmämäärityksiä ja eivät osaa pelkän sanallisen kuvauksen perusteella ennustaa miten järjestelmä vaikuttaa heidän työskentelyynsä. Heidän mukaansa esimerkiksi prototyypitys ja pilottiprojektit ovat tehokkaita tapoja tuoda käyttäjiä konkreettisesti mukaan suunnitteluun.

Jatkuvan osallistumisen haastavina puolina suunnittelijoille ja erityisesti kokemattomille suunnittelijoille voi olla se, että käyttäjien jatkuva osallistuminen ja vaatimusten muokkaaminen heidän toiveidensa mukaan lisää epävarmuutta projektin tavoitteista ja luo paineita sekä monimutkaistaa päätöksentekoa. Suunnittelijoiden on kuitenkin lopulta päätettävä palautteen pohjalta, miten järjestelmä toteutetaan ja mitä ehdotuksista huomioidaan. Käyttäjät esittävät usein paljon toivomuksia, joista kaikkia ei voida realistisesti toteuttaa ja toiveet ovat usein ristiriitaisia. (Kautz 2011, 224)

Lisäksi käyttäjien tuominen aktiivisesti mukaan suunnitteluun ja vallan jakaminen suunnittelupäätöksistä heidän kanssaan voi tuntua suunnittelijoille haastavalta. Kun käyttäjille esimerkiksi hyväksymistestien kautta annetaan todellista valtaa suunnittelupäätöksiin, voidaan joutua tekemään myös hyvin vaikeita päätöksiä. Kokonaisten suunnittelun osien peruuttaminen ja vaatimusten jatkuva muokkaaminen vaatii suunnittelijoilta joustavuutta ja sopeutumiskykyä. Asiaan pitäisi suhtautua niin, että jos suunnittelijat testaavat jotain menetelmää ja tiputtavat sen pois sopimattomana, on onnistuttu. Arvioinnin tuloksena suunnittelua voidaan muokata vastaamaan paremmin muuttuneita tarpeita tai jopa lopettaa, jos todetaan, että vaaditut

muutokset ovat liian suuria ja projekti ei ole enää järkevä. (Remenyi & Sherwood-Smith 1999, 28)

Suunnittelijoihin kohdistuvaa painetta jatkuvasti esitettyjen toiveiden takia voidaan jonkin verran ehkäistä sillä, että palautetta ei iteraatioiden välillä kerätä kuin tietyin sovituin väliajoin, jolloin suunnittelua voidaan tehdä sopivissa paloissa ennen sen altistamista uusille vaatimuksille. (Kautz 2011, 224) Yhteys käyttäjiin on tasapainoilua suunnitelurauhan säilyttämisen ja aktiivisen yhteydenpidon ja palautteen keräämisen välillä. Downing (2006, 48) korostaa sitä, että suunnittelijoiden pitää olla tarkkana, etteivät he anna periksi kaikille vaatimuksille, vaan punnitsevat tarkkaan mitkä ehdotukset kannattaa toteuttaa. Tämä tasapainoilu vaatimusten toteuttamisen kanssa aiheuttaa kokemattomille suunnittelijoille helposti paineita, varsinkin jos käyttäjät ovat heitä huomattavasti kokeneempia ja pyrkivät painostamaan suunnittelijoita omien näkemystensä toteuttamiseen.

Suunnitelurauhakaan ei tosin tarkoita kovin pitkiä taukoja palautteen keräämisessä ja suunnittelijoiden on opittava asiakaslähtöiseen ajatteluun. Esimerkiksi Kautzin (2011, 232) tutkimassa projektissa laajempien koko järjestelmän laajuisten iteraatioiden kesto oli noin kolmesta kuuteen viikkoa ja pienempien osakokonaisuuksien arviointi, muutospyyntöjen kerääminen käyttäjiltä tapahtui yleensä noin viikon välein ja lisäksi päivittäin kokoonnuttiin keskustelemaan vapaamuotoisemmin käyttäjien kanssa projektista. Silti vaikka Kautzin (2011, 229) tutkimassa projektissa käyttäjien palautteen kerääminen oli erittäin aktiivista, osa käyttäjistä toivoi vielä entisestään aktiivisempaa osallistumista. Tämä kertoo siitä, että palautteen keräämisen todella on oltava aktiivista ja nopeatahtista, että käyttäjien panosta voidaan hyödyntää tehokkaasti.

Tämä jatkuva käyttäjäpalautteen kerääminen vaatii suunnittelijoilta erilaisia taitoja kuin pääosin vain suunnitteluryhmän kesken tapahtuva suunnittelu. Iteratiivisen kehittämisen haasteena on pidetty sitä, että suunnittelijoilta vaaditaan paljon enemmän kommunikaatio-, ihmissuhde- ja koordinointitaitoja kuin lineaarisissa malleissa, koska käyttäjäyhteisö täytyy todella pystyä tuomaan mukaan kehittämistyöhön (Kautz 2011, 226). Kokemattoman projektiryhmän kannalta tämä voi olla ongelma. Kommunikointi- ja koordinointitaitojen määrä ei kuitenkaan välttämättä ole suoraan riippuvainen siitä, miten kokematon projektiryhmä on, mutta ainakin lyhyen aikaa organisaatiossa toimineille suunnittelijoille vaatimus jatkuvasta kommunikoin-

nista käyttäjien kanssa voi olla haastava ja viedä huomiota muusta kehittämisestä, jolloin projektin hallittavuus kärsii.

Toinen potentiaalinen ongelma iteratiivisuudessa on se, että se voi lisätä suunnittelijoiden pelkoa projektin mittakaavan hallitsemattomuudesta, koska käyttäjiltä saatava palaute lisää koko ajan järjestelmälle asetettuja vaatimuksia ja projekti lähtee helposti paisumaan (Center for Technology in Government 1998, 5). Myös Remenyi ja Sherwood-Smith (1999, 25) ovat esittäneet, että aktiivinen arviointi ja siitä saatu palaute, joka muokkaa vaatimuksia jatkuvasti, voi helposti saada projektin toteutuksen tuntumaan kohtuuttoman laajalta ja hallitsemattomalta.

Toisaalta iterointi jakaa suunnittelun kokonaisuuksia pienemmiksi, jolloin osien hallinnasta tulee helpompaa. Käyttäjien esittämien vaatimusten suuri määrä voi toki lisätä paineita järjestelmän kehittämiseen ja vaatii suunnittelijoilta selkeää linjaa, jolla päätöksiä toteutettavista muutoksista tehdään. Suuri ristikkäisten vaatimusten määrä vaikeuttaa projektin hallittavuutta. Palaute auttaa kuitenkin suunnittelijoita oppimaan koko ajan enemmän käyttäjien tarpeista ja lisää samalla järjestelmän laatua, koska siitä voidaan tehdä paremmin käyttäjien tarpeita vastaava. Hallittavuuden väheneminen on pienempi ongelma kuin se, että suunnitellaan järjestelmiä, jotka eivät tue liiketoimintoa ja ovat tarpeettomia.

Seuraavassa on vielä koottu yhteen vaikutuksia, joita työssä esitellyillä ketterän kehityksen sovellustavoilla ja niistä rakennetulla kokonaismallilla on suunnittelun hallittavuuteen erityisesti kokemattomien suunnittelijoiden kannalta.

Hallittavuutta lisääviä tekijöitä:

- virheiden hinta laskee, kun ne havaitaan nopeammin
- virheiden hinta laskee, kun niitä voidaan korjata helpommin jälkikäteen
- mahdollisuus tehokkaampaan reflektointiin jatkuvan palautteen pohjalta
- uusia ideoita voidaan testata tehokkaasti ja hylätä huonot ideat nopeasti
- vaiheittainen suunnittelu ja tietämyksen rakentaminen nopeassa tahdissa aiempien tulosten päälle edistävät suunnittelijoiden oppimista
- palautetta saadaan paljon ja säännöllisesti
- konkreettiset mallit tekevät palautteesta realistista

- palautteen lisäksi tiivis yhteys lisää asioiden oppimista käyttäjiltä
- käyttäjät voivat osallistua suunnitteluun aktiivisesti ja tukea sitä monipuolisemmin
- tiivis yhteys käyttäjiin tehostaa organisaation muutosten havaitsemista
- jatkuva yhteys käyttäjiin toimii tehokkaana tiedotuskanavana
- vaatimusmäärittelyitä voidaan tarkentaa joustavasti
- malli on luonteva pohja jatkuvalle kehittämiselle varsinaisen projektin loputtua
- pilottiprojektien toteuttaminen valmistaa järjestelmän käyttöönottoon

Hallittavuutta vähentäviä tekijöitä:

- mallin monimutkaisuus ja jakautuminen osiin vaikeuttavat suunnittelun hallintaa
- tarve mallin jatkuvalle soveltamiselle ja muokkaamiselle vaatii soveltamis- ja päätöksentekokykyä
- verrattuna lineaarisiin menetelmiin ketterät kehitysmenetelmät monimutkaistavat suunnittelua
- pilottiprojektien toteuttaminen vaatii erilaista osaamista ja niistä ei usein saada paljoa hyödyllistä tietoa
- vaatimusten joustava muokkaaminen vaikeuttaa projektin hallintaa
- vaatimusten jatkuva muuttuminen voi aiheuttaa suunnittelijoissa mittakaavan hallitsemattomuuden tunnetta
- suunnitteluvallan jakaminen käyttäjien kanssa vähentää kontrollointimahdollisuuksia
- käyttäjälähtöisyys lisää kommunikaatio- ja koordinointitaitojen merkitystä
- ketterät kehitysmenetelmät lisäävät suunnittelijoiden vastuuta ja itsenäisyyttä

Ketterän kehityksen menetelmien soveltamisen vaikutus suunnittelun hallittavuuteen ei siis ole yksiselitteinen. Vaikka ketterien menetelmien soveltamisella on monia myönteisiä vaikutuksia hallittavuuteen, kuten suunnittelijoiden kyky oppia tehokkaammin ja hyödyntää käyttäjien osaamista ja tietoa, ketterien menetelmien soveltaminen myös lisää osaltaan suunnittelijoihin kohdistuvaa painetta ja suunnittelun monimutkaisuutta. Hallittavuutta vähentäviin tekijöihin voidaan tosin jossain määrin vaikuttaa sillä, miten aktiivisesti ketteriä kehitysmenetelmiä sovelletaan suunnittelun eri vaiheissa ja miten niitä painotetaan.

6 PÄÄTELMÄT

Tarkastelin työssä, miten ketterän kehityksen menetelmät voisivat vaikuttaa sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun. Kehittämäni malli itsessään on työn päätulos vaikutusten arvioinnin ohella. Ketterien kehitysmenetelmien soveltaminen sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun vaikuttaa mallin rakentamisen pohjalta olevan paitsi mahdollista, myös monelta osin lupaava lähestymistapa verrattuna yleisesti käytettyihin lineaarisiin suunnittelumalleihin. Ketteriä kehitysmenetelmiä käsittelevässä kirjallisuudessa korostetaan usein sitä, että menetelmien tarkoitus on tehdä suunnittelusta joustavampaa ja nopeampaa. Vaikka joustavuuden ja nopeuden kriteerit ovat sähköisen asiakirjajärjestelmän suunnittelussa välillä ristiriitaisia, ketterillä kehitysmenetelmillä voidaan työn perusteella lisätä sekä nopeutta että joustavuutta painottaen niitä tarpeen mukaan erilaisissa tilanteissa ja projekteissa.

Asetin työssä joustavuuden tarkastelemiselle kriteeriksi erityisesti sen, miten hyvin mallin joustavuus tukee käyttäjien huomioimista suunnittelussa. Kirjallisuuden perusteella on perusteltua väittää, että ketterät kehitysmenetelmät mahdollistavat lineaarisiin menetelmiin verrattuna käyttäjälähtöisemmän tavan suunnitella järjestelmiä. Myös nyt rakennettuun malliin käyttäjien mukaan tuominen suunnitteluun ja palautteen aktiivinen kerääminen suunnittelun joka vaiheessa istuvat luontevasti. Ketteriin kehitysmenetelmiin pohjautuvassa mallissa erityisesti iteratiivisuus ja suunnittelua konkretisoivat välineet, kuten prototyypit ja pilottiprojektit edistävät käyttäjien sitouttamista suunnitteluun. Iteroinnin avulla palautetta voidaan kerätä joustavasti ja jatkuvasti toisin kuin lineaarisissa malleissa, joissa käyttäjien todellinen sitouttaminen suunnitteluun on vaikeasti toteutettavissa suunnittelun yksisuuntaisuuden ja laajoihin kokonaisuuksiin jakautumisen takia.

Mallin rakentamisen jälkeen pohdin sitä, miten asetetuilla kriteereillä suunniteltu malli vaikuttaa kokemattoman suunnitteluryhmän mahdollisuuksiin hallita suunnittelua. Ketterien kehitysmenetelmien soveltaminen tämän mallin mukaisesti on kirjallisuuden perusteella joiltain osin potentiaalisesti hyödyllistä, mutta ongelmiakin on. Keskeisimpiä hyötyjä ovat lisääntynyt yhteys käyttäjiin ja mahdollisuus peilata suunnittelua käyttäjien palautteeseen sekä hyödyntää heidän osaamistaan suunnittelussa. Lisäksi mallin vahva iteratiivisuus paitsi jakaa suunnittelua hallittavampiin kokonaisuuksiin, myös mahdollistaa tehokkaamman oppimisen, kun ai-

emmista iteraatioista saatua osaamista ja tietoa voidaan soveltaa käytännössä nopealla aikataululla seuraavissa iteraatioissa. Myös voimakas osittaminen jakaa suunnittelua hallittavampiin kokonaisuuksiin ja auttaa erityisesti kokemattomia suunnittelijoita pitämään suunnittelun hallinnassa.

Malli aiheuttaa kokemattoman suunnitteluryhmän kyvylle hallita suunnittelua myös monia haasteita. Vaikka lisääntynyt yhteys käyttäjiin tukee oppimista ja osittain myös projektin hallintaa, se edellyttää suunnittelijoita uudenlaisia taitoja ja asennetta kommunikaatioon ja suunnitteluun. Suunnittelijoiden on kyettävä jakamaan valtaa käyttäjien kanssa ja otettava heidän toiveensa aktiivisesti huomioon. Tästä seuraa erityisesti kokemattomille suunnittelijoille helposti painetta, kun käyttäjät esittävät koko ajan parannusehdotuksia, jotka ovat usein ristiriitaisia. Jatkuva vaatimusten muuttuminen ja ristiriitaisten vaatimusten punninta lisää suunnittelun monimutkaisuutta, mikä puolestaan vähentää hallittavuutta. Samoin voimakas osittaminen paitsi tekee yksittäisistä osista hallittavampia myös monimutkaistaa kokonaisuuden hallintaa, kun suunnittelun ohella pitää hallita myös eri osakokonaisuuksia suhteessa toisiinsa ja koko suunnitteluun. Myös yleisesti nyt esitelty malli on monimutkaisempi kuin lineaariset mallit, joissa suunnittelun eteneminen vaiheesta toiseen on selkeämpää. Lisäksi jotkut mallin osa-alueet, kuten pilottiprojektit, asettavat kokemattomille suunnittelijoille haasteita, koska varsinaisen suunnittelun ohella ne vaativat erityisosaamista.

Tulosten pohjalta asettamani hypoteesi siitä, että ketterien kehitysmenetelmien soveltaminen sähköisen asiakirjajärjestelmän suunnitteluun nopeuttaa välitavoitteiden saavuttamista ja joustavuutta pitää tutkimuksen perusteella mielestäni paikkaansa. Joustavuuden osalta ero lineaarisiin malleihin on selvä, sillä iterointiin perustuva suunnittelumalli sallii huomattavasti joustavamman liikkumisen suunnittelun vaiheissa ja suunnittelun kehittämisen vaiheittain. Mallin pohjalta on myös mahdollista kerätä iteroinnin ja erilaisten testaamis- ja palautteenkeräämismekanismien avulla lineaarisia malleja tehokkaammin palautetta ja tuoda käyttäjät mukaan suunnitteluun. Myös välitavoitteiden saavuttaminen nopeutuu, koska järjestelmää ei pyritä suunnittelemaan kerralla valmiiksi, vaan se jakautuu erilaisten selkeiden välitavoitteiden toteuttamiseen ja suunnittelun jatkamiseen niiden pohjalta. Toisaalta suunnittelun kokonaisnopeuden pieneneminen ei ole niin selvää. Se ei ollut työssä tarkemman tarkastelun kohteena, mutta ketterien kehitysmenetelmien soveltamisessa on riski, että suunnittelun kokonaiskestoa on vaikeaa arvioida ja se kasvaa helposti. Tätä aikatauluongelmaa voidaan tosin hallita monilla ketterän kehityksen pohjalta kehitetyillä menetelmillä, mutta niitä ei tässä työssä käsitelty.

Hallittavuuden osalta hypoteesin toteutuminen on epäselvempää. Mallin hyödyntämisessä on hallittavuuden kannalta sekä etuja että ongelmia. Malli todennäköisesti edistää suunnittelijoiden oppimista, mutta hallittavuus voi kärsiä esimerkiksi projektin hajaantumisen ja lisääntyneiden käyttäjävaatimusten takia.

Kokemattomien suunnittelijoiden tuominen mukaan ketterien kehitysmenetelmien soveltamiseen tuo esiin eroja aiemmassa kirjallisuudessa esitettyihin ketterän kehityksen vahvuuksien ja heikkouksien pohdintaan. Vaikka kirjallisuudessa esitetyt ketterän kehityksen menetelmät vaikuttavat olevan järkevästi sovellettavissa sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun, oletuksena kirjallisuudessa on useimmiten se, että suunnittelua toteuttavat kokeneet ja tarvittavat taidot omaavat suunnittelijat. Näin ei käytännössä kuitenkaan useinkaan ole ja suunnittelijoiden kokemattomuus on tärkeä tekijä erilaisten menetelmien soveltamisen järkevyyttä arvioitaessa. Pääosin toki vaikuttaa siltä, että kirjallisuudessa esitetyt vahvuudet ketterien kehitysmenetelmien soveltamisesta ja nopeuden sekä joustavuuden korostaminen pätevät suunnittelijoihin kokemuksesta riippumatta. Edellisissä kappaleissa esitetyt mallin vaikutukset, jotka korostuvat erityisesti kokemattomilla suunnittelijoilla, osoittavat kuitenkin, että kokemattomuus on tekijä, joka ketteriä kehitysmenetelmiä ja suunnittelua yleensä käsiteltäessä syytä huomioida paremmin.

Tutkimuksen asetelma täydentää kuvaa ketterien kehitysmenetelmien soveltamisen osalta kahdella tavalla. Ensinnäkin se toimii esimerkkinä siitä, miten ketteriä kehitysmenetelmiä voitaisiin soveltaa asiakirjahallinnan suunnittelussa. Aiemmassa asiakirjahallinnan tutkimuskirjallisuudessa ketterät kehitysmenetelmät on muutamaa yksittäistä kirjoitusta lukuun ottamatta jätetty huomiotta. Verrattuna näihin muutamiin aiempiin kirjoituksiin, tässä työssä pohditaan ketterien kehitysmenetelmien soveltamista asiakirjahallinnan suunnitteluun konkreettisemmalla menetelmien käytännön soveltamisen tasolla. Tarkoituksena on ollut esitellä paitsi menetelmien soveltamista sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun, myös yleisemmin sitä, miten menetelmiä voi soveltaa asiakirjahallinnan suunnittelutyössä.

Ketterien kehitysmenetelmien asiakirjahallintaan soveltamisen esittelyn ohella toinen keskeinen aiemmassa tutkimuksessa vain vähän käsitelty tekijä on kokemattomien suunnittelijoiden käsittely. Aihetta ei ole hallintotieteisiin ja tietojenkäsittelytieteisiin painottuneen kartoitukseni perusteella käsitelty paljoa, mutta erityisesti asiakirjahallinnan kirjallisuudesta tämä näkökulma näyttäisi puuttuvan kokonaan. Jostain syystä asiakirjahallinnan ohjeistuksissa suunnit-

telijoiden osaaminen sivuutetaan yleensä kokonaan ja oletetaan aina optimaalinen kaiken osaamisen omaava suunnitteluryhmä, vaikka tämä ei yleensä ole realistinen lähtökohta. Työ tuo osaltaan tätä näkökulmaa esille ja pohtii, voisiko ketterien kehitysmenetelmien kaltainen joustavampi käyttäjälähtöinen suunnittelutapa edistää myös kokemattomien suunnittelijoiden mahdollisuuksia hallita asiakirjahallinnan projekteja. Mielestäni ketterien kehityksen menetelmien soveltaminen vaikuttaa tämän työn perusteella lupaavalta keinolta kokemattomien suunnittelijoiden tukemisessa, mutta ilman käytännön kokemuksia ja empiirisiä testejä mallin toimivuudesta tätä ei voi tutkia tehokkaasti.

Kokemattomien suunnittelijoiden haasteet yleisesti suunnittelutyössä ja erityisesti asiakirjahallinnan suunnittelussa vaatisivat enemmän tutkimusta, koska osaamisen puutteen ongelma on todennäköisesti laaja ja vaikuttaa projektien toteuttamiseen merkittävästi. Kokemattomuus ei myöskään rajoitu vain asiakirjahallinnan töistä yleensä vähän kokemusta omaaviin uusiin työntekijöihin, vaan myös niihin joilla on puutteellinen osaaminen jostain asiakirjahallinnan kehittämisestä, kuten asiakirjahallinnan sähköistämisestä. Tutkimuskirjallisuuden pohjalta vaikuttaa siltä, että perinteiset lineaariset menetelmät eivät pysty vastaamaan tehokkaasti kokemattoman projektiryhmän tarpeisiin oppia ja kehittyä suunnittelijoina projektin aikana, eivätkä tue heitä riittävästi projektien hallitsemisessa.

Koska aukot näiden molempien näkökulmien suhteen ovat asiakirjahallinnan kirjallisuudessa suuret, potentiaalisia jatkotutkimuksen aiheita on paljon. Tässä työssä pystyin käsittelemään näitä kahta laajaa aihetta vain suppeasti, koska työn laajuuden rajaamiseksi rajasin aiheiden käsittelyä voimakkaasti. Ensinnäkin esitelty esimerkkimalli on nimenomaan vain esimerkki. Samalla kaavalla ja asettamalla mallin rakentamisen kriteerit ja mallin pohjalta tutkittavan ilmiön eri tavalla, olisi mahdollista luoda lukuisia erilaisia toteutustapoja. Mallin rakentamisessa voisi esimerkiksi korostaa laatunäkökulmaa tai riskienhallintaa nopeuden ja joustavuuden sijaan. Vaikutusten arvioinnissa voisi kääntää tämän työn asetelman toisinpäin ja tutkia eri tavalla rakennetun mallin vaikutusta suunnittelun nopeuteen ja joustavuuteen. Tässä mallissa otettiin laajempi lähestymistapa ketterien kehitysmenetelmien soveltamiseen, mutta käytännössä menetelmiä voisi soveltaa myös suppeammin tai vielä laajempiin suunnittelun kokonaisuuksiin. Menetelmiä voidaan tarpeen mukaan painottaa hyvin eri tavoin riippuen tarkasteltavasta projektista ja sen vaatimuksista.

Mielenkiintoisin tapa jatkaa tutkimusta olisi jonkinlaisen empiirisen kokeen toteuttaminen mallin pohjalta ja mallin toimivuuden testaaminen käytännössä. Työssä sovelletussa Evansin ja Rouchen (2006, 318–320) mallissa kirjallisuuden pohjalta rakennetun suunnittelumallin testaaminen ja vaikutusten arviointi tapahtuu käytännössä rakentamalla testijärjestelmä ja testaamalla sitä todellisilla käyttäjillä. Nyt rakennettua mallia voisi testata esimerkiksi jonkinlaisella koeasetelmalla, jossa tutkittaisiin miten malli vaikuttaa sitä hyödyntävien suunnittelijoiden työskentelyyn ja verrata sitä vaikkapa toiseen ryhmään, joka toteuttaa suunnittelua lineaarisilla menetelmillä. Esimerkiksi Alavi (1984) on käyttänyt prototyyppien arvioinnissaan juuri tämänkaltaista tutkimustapaa, jossa hän vertasi prototyyppejä käyttäneen suunnitteluryhmän tuloksia ja menetelmiä lineaarisella suunnittelumallilla työskennelleisiin. Toinen mahdollisuus olisi soveltaa mallia todellisessa sähköisen asiakirjahallintajärjestelmän suunnittelussa jossain organisaatiossa ja kerätä tämän pohjalta esimerkiksi haastatteluilla kokemuksia mallin toimivuudesta tai pyrkiä kartoittamaan käyttäjien osallistumista suunnitteluun tai sitä, miten malli vaikuttaa kokemattomien suunnittelijoiden työn hallittavuuteen.

Mallin käytännön testaamisen puute on myös suurin puute työn tutkimusasetelmassa. Työn lähestymistapa tarkasteluun pohjautui kirjallisuuden ja omien kokemusten soveltamiseen ja oli siten hyvin teoreettinen. Tällaisenaan ketterän kehityksen menetelmien soveltaminen sähköisen asiakirjajärjestelmän suunnitteluun on enemmän valistunut arvaus, koska mallin toimivuutta ei testattu empiirisesti. Pidän lähestymistapaa kuitenkin perusteltuna tämänlaajuisessa työssä, koska mallin rakentaminen käytännössä olisi työmäärältään ollut kohtuuton ja testiorganisaation löytäminen haastavaa. Tämänkaltainen yleisempi ketterän kehityksen menetelmien soveltamisen vaihtoehtoja pohtiva ja kartoittava tutkimus oli mielestäni perusteltu myös siksi, että aiempaa kirjallisuutta aiheesta ei asiakirjahallinnan alueelta käytännössä ole. Tämän tutkimuksen pohjalta empiiristen testien toteuttaminen olisi jatkossa realistisempaa.

Työn rajauksessa ongelmana oli se, että tällä tavalla rajattuna tietosisällön suunnittelu jää irralliseksi muusta järjestelmän suunnittelusta ja laajemmin sähköisen asiakirjajärjestelmän toteutuksen kokonaisuudesta, johon kuuluvat esimerkiksi ennakkokartoitukset, vaatimusmäärittelyt ja järjestelmän tekninen toteuttaminen. Toisaalta tällä tavalla rajattuna pystyin tarkastelemaan ketterien kehitysmenetelmien soveltamista lähempänä konkreettista käytännön suunnittelun tasoa, mikä oli mielestäni perusteltua varsinkin, kun mallin pohjalta tarkastelin suunnittelun hallittavuutta suunnittelijoiden näkökulmasta, en esimerkiksi johdon osallistumista. Laajempi näkökulma olisi kuitenkin voinut selkeyttää käsittelyä ja auttanut ymmärtä-

mään tietosisällön limittymisen laajempaan suunnittelukokonaisuuteen. Koin kuitenkin tarpeelliseksi rajoittaa tarkastelua mahdollisimman pieneen osaan käsittelyn laajuuden pitämiseksi mielekkäänä. Tämän rajauksen etuna oli myös se, että olen itse toiminut suunnittelijana vastaavanlaisessa suppeammassa projektissa ja tämä auttoi arvioimaan suunnitteluratkaisujen toimivuutta.

Työtä olisi kuitenkin voinut rajata vielä tiukemmin mallin käsittelemän kokonaisuuden osalta. Esimerkiksi keskittyminen vain yhteen ketterän kehityksen menetelmään, kuten iterointiin olisi keventänyt mallin rakentamista ja jättänyt enemmän tilaa erilaisten ketterän kehityksen näkökulmien käsittelyyn ja vaikutusten pohdintaan. Tällä tavalla ketterän kehityksen perusmenetelmät olisivat kuitenkin jääneet laajemmin esittelemättä ja malli olisi toiminut huonommin esimerkkinä menetelmien soveltamisesta asiakirjahallintaan.

Rajaus sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun oli mielestäni perusteltu. Tässäkin laajempi näkökulma, esimerkiksi koko asiakirjajärjestelmän suunnittelun mukaan ottaminen olisi ollut mielenkiintoista, mutta työ olisi tällä tavalla toteutettuna kasvanut liian laajaksi. Tässä täytyy huomioida myös se, että työn tarkoituksena oli vähintään yhtä paljon kuin suunnitella vaihtoehtoinen malli sähköisen asiakirjajärjestelmän tietosisällön suunnitteluun, myös esitellä miten ketteriä kehitysmenetelmiä yleensä voidaan soveltaa asiakirjahallinnan suunnittelussa. Ketteriä kehitysmenetelmien vahvuus on kuitenkin se, että niitä voidaan sinällään soveltaa hyvin joustavasti erilaisiin ja erikokoisiin projekteihin ja suunnitteluihin.

Työssä esiteltyjä menetelmiä voisi joko tämän mallin pohjalta tai poimimalla yksittäisiä menetelmiä tarpeen mukaan soveltaa todennäköisesti onnistuneesti esimerkiksi asiakirjahallinnan kokonaissuunnitteluun ja toteutukseen tai vaikkapa vain yksittäisen sähköisen asiakirjahallintajärjestelmän tietosisällön suunnittelun vaiheeseen. Tässäkin mallissa esitellystä sähköisen asiakirjajärjestelmän suunnittelusta ketterien kehitysmenetelmien avulla voi havaita sen, miten malli jakaantuu ikään kuin sisäkkäisiin projekteihin. Näitä eri kerroksia, kuten yksittäisen vaiheen suunnittelua, osakokonaisuuden suunnittelua, koko tietosisällön suunnittelua ja laajemmin koko asiakirjajärjestelmän toteuttamista voidaan ajatella pääosin vain mittakaavaltaan erilaajuisina suunnittelun kokonaisuuksina, joista kaikkiin kuitenkin kuuluvat samankaltaiset suunnittelun vaiheet ja niihin voidaan soveltaa samankaltaisia ketterän kehityksen menetelmiä tai hieman muokaten tässä työssä esiteltyä suunnittelumallia.

Koska työssä sovelletut ketterät kehitysmenetelmät ovat enemmän tapoja toteuttaa suunnittelua ja tietynlainen filosofia kehittämiselle, mallissa esitettyjä menetelmiä ja mallin lähestymistapoja voi ajatella paitsi skaalattaviksi pienempiin tai suurempiin sähköisen asiakirjajärjestelmän suunnittelukokonaisuuksiin, myös sovellettavaksi muissa asiakirjahallinnan suunnitteluun liittyvissä projekteissa. Esimerkiksi niinkin erilaiset asiakirjahallinnan työt kuin arkiston järjestäminen tai asiakirjakokonaisuuden kuvailu voisivat hyötyä ketterän kehityksen menetelmistä, kuten iteroinnista ja prototyyppien hyödyntämisestä. Laajemmin työssä esitellyt ketterän kehityksen keskeiset menetelmät ja ketterän kehittämisen filosofia voivat olla käytännössä osa mitä tahansa suunnittelua, kun halutaan lisätä suunnittelun joustavuutta, nopeutta ja käyttäjälähtöisyyttä. Esimerkiksi Minna Niemi-Grundström (2012) on korostanut sitä, miten ketterän kehityksen periaatteita voidaan soveltaa organisaatioissa kaikessa suunnittelussa ja kehittämisessä, ei pelkästään järjestelmien suunnittelussa.

Tässä työssä näkökulma eli enemmän konkreettisten ketterän kehityksen menetelmien soveltamisessa. Yleisempää filosofista ja periaatteellista puolta käsiteltiin vain vähän, vaikka ketterien ketterissä kehitysmenetelmissä asenne ja yleinen lähestymistapa kaikkeen toimintaan ja suunnitteluun ovat usein vähintään yhtä tärkeitä. Yksi näkökulma jatkotutkimukseen voisi olla laajempi pohdinta siitä, miten ketterän kehittämisen filosofiaa ja periaatteita voisi soveltaa esimerkiksi erilaisien asiakirjahallinnan suunnittelua ja toteuttamista ohjaavien ohjeistusten laatimisessa tai jokapäiväisessä suunnittelutyössä organisaatioissa.

Toisaalta laajemman näkökulman vastapainoksi ketterien kehitysmenetelmien soveltamisessa ei välttämättä tarvitse olla kysymys kokonaisvaltaisesta ajattelutavan muutoksesta, vaan niiden soveltaminen voi olla myös käytännönläheistä ja enemmän asenne suunnitteluun tai työskentelyyn kuin kokonaan erilainen lähestymistapa. Tähän liittyen ketterien kehitysmenetelmien voi soveltaa erilaisiin pieniin projekteihin ja olisi myös mahdollista tutkia niiden hyödyntämistä konkreettisesti jossain pienessä projektissa, kuten esimerkiksi aiemmin mainituissa arkiston järjestämisessä tai asiakirjojen kuvailussa. Ketterien kehitysmenetelmien avulla voidaan paitsi nopeuttaa ja joustavoittaa suunnittelua, myös konkreettisesti edistää monia sellaisia suunnittelun tavoitteita, kuten jatkuvaa kehittämistä, jotka yleensä mainitaan erilaisissa ohjeistuksissa tavoitteiksi, mutta joita on käytännössä vaikea toteuttaa, koska projektit toteutetaan laajoina harvoin toistuvina projekteina.

Työn rajauksessa ongelmia liittyy myös kriteereihin, joilla sovelsin menetelmiä malliin. Kuten aiemmin totesin, nopeuden ja joustavuuden kriteerit tuntuivat toimivan kyllä ihan hyvin ja auttoivat järjestelmällisemmin muotoilemaan mallia ja valitsemaan toteutustapoja. Myös asiakaslähtöisyys toimi selkeänä lähtökohtana. Kuitenkin erityisesti laatunäkökulman poisjättäminen tuntui puutteelta. Koska valinnan kriteereinä olivat vain nopeus ja joustavuus, jouduin jättämään mallista pääosin pois käyttäjien toteuttaman suunnittelun jatkuvan arvioinnin, jolla pyritään varmistamaan järjestelmän laatu, johon kuuluu esimerkiksi järjestelmän vastaavuus käyttäjien tarpeisiin. Tämän näkökulman huomiointi mallissa olisi mielestäni perusteltua, koska laadun varmistaminen on tärkeä osa toimivan ja hyödyllisen järjestelmän suunnittelua. Laatunäkökulma voisi olla hyödyllinen myös kokemattomille suunnittelijoille, sillä se auttaisi keräämään järjestelmällisemmin palautetta ja varmistamaan, että suunnittelijat eivät unohda käyttäjien tarpeita ja suunnittelun tasoa voidaan jatkuvasti tarkkailla ja arvioida.

Laatunäkökulman ohella olisi ollut mahdollista asettaa mallin rakentamisen kriteeriksi riskienhallinnan, mikä myös on hyödyllinen näkökulma, joka auttaisi esimerkiksi muuttamaan suunnittelun suuntaa tai hylkäämään huonoja suunnittelutoteutuksia, jos niiden tuottama hyöty uhkaa jäädä vähäiseksi. Toisaalta tässäkin oli pakko rajata kriteereitä työn pitämiseksi hallittavana. Jos kuitenkin nyt määrittelisin mallin rakentamisen kriteereitä, valitsisin ainakin laatunäkökulman joko aiempien kriteerien oheen tai jonkun nyt käsitellyn kriteerin tilalle.

LÄHTEET

Alavi, Maryam 1984. An assessment of the prototyping approach to information systems development. Communications of the ACM, vol. 27 no. 6, 556–563.

Arkistolaitos 2010. eAMS-käyttöönottosuunnitelmaohje. <http://www.arkisto.fi/uploads/eAMS-käyttöönottosuunnitelma18082010.pdf>, luettu 2.5.2012.

Bansler, Jørgen & Havn, Erling 2010. Pilot implementation of health information systems: Issues and challenges. International Journal of Medical Information, vol. 79 no. 9, 637–648.

Boehm, Barry & Gray, Terence. & Seewaldt, Thomas. 1984. Prototyping vs. specifying: A multi-project experiment. Esitelmä the 7th International Conference on Software Engineering -konferenssissa Floridan Orlandossa.

Center for Technology in Government 1998. A Survey of System Development Process Models. <http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf>, luettu 28.9.2012.

Collyer, Simon & Warren, Clive & Hemsley, Bronwyn & Stevens, Chris 2010. Aim, Fire, Aim—Project Planning Styles in Dynamic Environments. Project Management Journal, September 2010, 108–121.

Di Biagio, Maria Luisa & Ibiricu, Bernice 2008. A balancing act: learning lessons and adapting approaches whilst rolling out an EDRMS, European Central Bank, Frankfurt am Main, Germany. Records Management Journal, vol. 18 no. 3, 170–179.

Downing, Lynette 2006. Implementing EDMS. Putting People First. The Information Management Journal, vol. 40 no. 4, 44–50.

Evans, Joanne & Rouche, Nadav 2006. Utilizing Systems Development Methods in Archival Systems Research: Building a Metadata Schema Registry. Archival Science, vol. 4, no. 3–4, 315–334.

Gerrard, Paul 2009. The W-Model. <http://www.gerrardconsulting.com/?q=node/531>, luettu 27.9.2012.

Glass, Robert 1997. Pilot Studies: What, Why, and How. Journal of Systems and Software, vol. 36, no. 1, 85–97.

Goldman, Steven & Roger Nagel & Kenneth Preiss 1995. Agile Competitors and Virtual Organizations – Strategies for Enriching the Customer Needs. Van Nostrand Reinhold, New York.

Gunnlaugsdottir, Johanna 2006. The Implementation and Use of ERMS. A Study in Icelandic Organizations. Acta Universitatis Tamperensis 1185. Tampereen yliopistopaino Oy, Tampere.

Gunnlaugsdottir, Johanna 2008. As you sow, so you will reap: implementing ERMS. Records Management Journal, vol. 18 no. 1, 21–39.

Highsmith, Jim 2004. Agile Project Management: Creating Innovative Projects. Pearson Education, Boston.

Jain, Rashmi & Chandrasekaran, Anithashree 2009. Rapid System Development (RSD) Methodologies: Proposing a Selection Framework. Engineering Management Journal, vol. 21 no. 4, 30–35.

Johnston, Gary 2004. Applying IS development techniques to improve the quality of Records Management systems.
<http://www.audata.co.uk/component/option,com_docman/task,doc_download/gid,2/>, luettu 26.9.2012.

Johnston, Gary 2005. An alternative model for the design and implementation of records management systems. RMS bulletin, June 2005, 13–17.

Kansallisarkisto 2012. AMS-opas. <http://www.ams-opas.fi/>. Luettu 2.5.2012.

Karlström, Daniel & Runeson, Per 2005. Combining Agile Methods with Stage-Gate Project Management. IEEE Software, vol. 2 no. 3, 43–49.

Kautz, Karlheinz 2011. Investigating the design process: participatory design in agile software development. Information technology & People, vol. 24 no. 3, 217–235.

Leyborne, Stephen 2009. Improvisation and agile project management: a comparative consideration. International Journal of Managing Projects in Business, vol. 2 no. 4, 519–535.

Liou, Frank 2008. Rapid Prototyping and engineering applications: a toolbox for prototype development. CRC Press, Boca Raton.

Malcolm, Eva 2001. Requirements acquisition for rapid application development. Information & Management, vol. 39 no. 2, 101–107.

McConnell, Steve 1996. Rapid development: taming wild software schedules. Microsoft Press, Redmond.

Najjar, Lawrence 1990. Rapid Prototyping. <http://www.lawrence-najjar.com/papers/Rapid_prototyping.html>, luettu 11.10.2012.

National Archives of Australia 2001. DIRKS: A Strategic Approach to Managing Business Information. <http://www.naa.gov.au/records-management/publications/DIRKS-manual.aspx>, luettu 14.4.2012.

Niemi-Grundström, Minna 2012. Ketterät menetelmät kirjaston kehittämisessä. Luento Tampereen yliopistossa 4.5.2012.

Remenyi, Dan & Sherwood-Smith, Michael 1999. Maximise information systems value by continuous participative evaluation. Logistics Information Management, vol. 12 no. 1–2, 14–31.

Roodenriis, Ewald 2009. The W-model. <http://www.testingthefuture.net/2009/09/the-w-model/>, luettu 26.9.2012.

Rzevski, George 1984. Prototypes versus pilot systems: strategies for evolutionary information system development. Julkaistu teoksessa R. Budde, K. Kuhlenkamp, L. Mathiassen, L. Züllighoven (toim.). Approaches to Prototyping. Springer, Heidelberg, 341–356.

Senge, Peter 1994. The Fifth Discipline: The Art & Practice of the Learning Organization. Currency Doubleday, New York.

Smyth, Zoë 2005. Implementing EDRM: has it provided the benefits expected? Records Management Journal, vol. 15 no. 3, 141–149.

Snyder, Charles & Cox, James 1985. A Dynamic Systems Development Lifecycle Approach: A Project Management Information System. Journal of Management Information Systems, vol. 2 no. 1, 61–76.

Suomen standardisoimisliitto 2007. SFS-ISO 15489-1, Tieto ja dokumentointi. Asiakirjahallinto, osa 1: yleistä. Suomen standardisoimisliitto, Helsinki.

Tikka, Jarmo 2010. Projektin toteuttamisen ja tietosisällön tuottamisen haasteet Kuopion ja Joensuun yliopistojen asiakirjahallinnan kehittämistyössä elokuusta 2009 vuodenvaihteeseen 2010. Ylemmän arkistotutkinnon syventävä tutkielma, Itä-Suomen yliopiston historia- ja maantieteiden laitos.

Upward, Frank 2000. Modelling the Continuum as paradigm shift in recordkeeping and archiving processes, and beyond - a personal reflection. Records Management Journal, vol. 10 no. 3, 115–139.

Vänttinen, Katri 2012. Asiantuntijat palveluorganisaatiossa, johtamisen näkökulmia muutoksessa. Luento Tampereen yliopistossa 20.4.2012.

Wilkins, Linda 2009. Achieved and tangible benefits lessons learned from a landmark EDRMS implementation. Records Management Journal, vol. 19 no. 1, 37–53.

Xie, Shane & Tu, Yiliu 2011. Rapid One-of-a-kind Product Development: Strategies, Algorithms and Tools. Springer, London.